Developing an On-demand Cloud-based Sensing-as-a-Service System for Internet of Things

Article *in* Journal of Computer Networks and Communications · August 2016

6 authors, including:

Jerry Gao
San Jose State University
172 PUBLICATIONS **2,327** CITATIONS

Article

Developing an On-demand Cloud-based Sensing-as-a-Service System for Internet of Things

Mihui Kim [1], Mihir Asthana[2], Siddhartha Bhargava[2], Kartik Krishnan Iyyer[2], Rohan Tangadpalliwar[2], and Jerry Gao[2,3,*]

[1] Department of Computer Science & Engineering, Computer System Institute, Hankyong National University/327, Jungang-ro. Anseong-si, Gyeonggi-do, Republic of Korea 456-749; mhkim@hknu.ac.kr
[2] Computer Engineering Department, San Jose State University, One Washington Square, San Jose, CA 95192, USA; (mihir.asthana, siddhartha.bhargava, kartikkrishnan.iyyer, rohan.tangadpalliwar, jerry.gao)@sjsu.edu
[3] Taiyuan University of Science and Technology, Taiyuan, China
* Correspondence: jerry.gao@sjsu.edu; Tel.: +1- 408-924-3904

Abstract: The increasing number of Internet of Things (IoT) devices with various sensors has resulted in a focus on Cloud-based sensing-as-a-service (CSaaS) as a new value-added service, e.g., providing temperature-sensing data via a cloud computing system. However, the industry encounters various challenges in the dynamic provisioning of on-demand CSaaS on diverse sensor networks. We require a system that will provide users with standardized access to various sensor networks and a level of abstraction that hides the underlying complexity. In this study, we aim to develop a cloud-based solution to address the challenges mentioned earlier. Our solution, SenseCloud, includes a ***sensor virtualization*** mechanism that interfaces with diverse sensor networks, a ***multi-tenancy*** mechanism that grants multiple users access to virtualized sensor networks while sharing the same underlying infrastructure, and a ***dynamic provisioning*** mechanism to allow the users to leverage the vast pool of resources on demand and on a pay-per-use basis. We implement a prototype of SenseCloud by using real sensors and verify the feasibility of our system and its performance. SenseCloud bridges the gap between sensor providers and sensor data consumers who wish to utilize sensor data.

Keywords: Cloud-based Sensing-as-a-Service (CSaaS); Cloud-based Internet of Things (IoT); Big Data Sensing; Sensor Cloud; Cloud Computing; Prototype Implementation

1. Introduction

As numerous devices and sensors get connected to the Internet, Internet of Things (IoT) is becoming a key topic of interest. Cisco predicts that 50 billion devices will be connected to the Internet by 2020. These devices and sensors will generate approximately 403 zetta bytes of data per year by 2018. In this fast-paced environment, the management of these devices, the networks, and the generated data is vital. The management and provisioning of such sensor devices and data opens doors to new business opportunities and poses new challenges. Industry and academia must manage these interconnected devices and exploit the opportunity presented by the extremely large amount of data generated. However, the huge investment and high maintenance cost of sensor network infrastructure prevents users from building their own IoT systems and web applications that utilize sensor data.

Thus, cloud-based sensing-as-a-service (CSaaS) appears as a new service paradigm; its system architecture is shown in Figure 1 [1,2]. Sensor providers supply sensor data on mobile devices or on sensor networks through operations to subscribe to and publish sensing data. Big data are saved and processed on the CSaaS infrastructure. Sensor consumers utilize the sensing data from the cloud system on demand. The following important and specific challenges must be carefully addressed in the design and implementation of a CSaaS system [2]. First, the system must be generic and must hide the underlying complexity such that it can support various opportunistic and participatory

sensing applications (which may even involve a large variety of sensors) and incurs very little overhead when launching a new sensing application and service. Second, the system can provide efficiency and sustainable scalability while using the same underlying infrastructure and ensuring the lowest cost-of-service delivery for each incremental consumer. Third, the system can provide the service dynamically to allow consumers to leverage the vast pool of resources on demand.

Several models have been proposed in order to address some of these challenges in sensor networks. Service-oriented architecture (SOA) approaches [3–5] integrate wireless sensor networks (WSNs) and leverage the widespread use of WSNs. Service-centric models [6–11] focus on the services provided by a WSN as a service provider. Approaches using crowd-sourced mobile devices as sensors have been proposed [12–14]; they utilize the sensors of existing mobile devices to fulfil user requests. Integrated approaches to share the sensing data on existing sensor networks [15–17] provide efficient sharing mechanisms among multiple applications. However, these studies have not resolved the challenges related to the engagement and connectivity for diverse sensor networks, multi-tenancy considering both sensor providers and consumers in CSaaS, and on-demand big data sensing service. Further, these studies have not demonstrated a prototype implementation of the platform with real sensors.

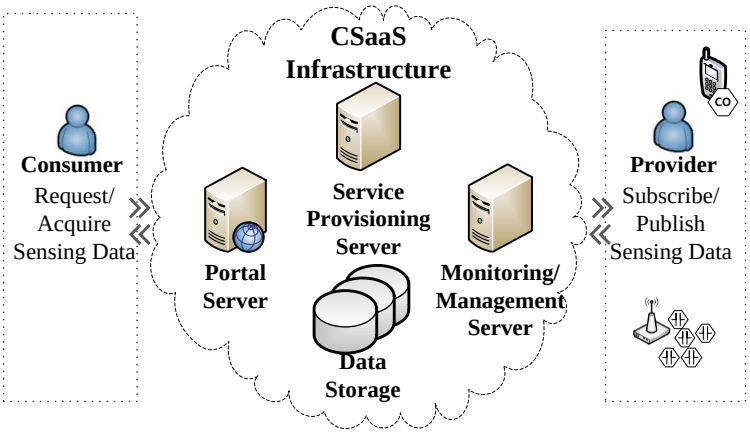


Figure 1. Service Architecture on CSaaS Infrastructure

In this study, we develop a CSaaS system called SenseCloud. The system aims to create a universal cloud platform with the following features: 1) It engages and manages various sensors on IoT devices by using the virtualization layer; 2) it gathers data from diverse sources such as sensors and sensor networks for useful and predictive analysis on a cloud system; 3) it provides access to real-time and historical data for analysis on an intuitive and feature-rich web interface; 4) it gives multiple users access to virtualized sensor networks while sharing the same underlying infrastructure; 5) it provides easy and standardized sensing data service for consumers and third-party applications dependent on data, and 6) it allows dynamic provisioning for users to leverage the vast pool of resources on demand. We implement a prototype of SenseCloud with real sensors. Then, we evaluate the operation and feasibility of our system and verify the system performance in terms of request time, load balancing, and scalability.

The contributions of this study are as follows:

- Design of a CSaaS platform with virtualization, multi-tenancy, and dynamic provisioning.
- Sensor virtualization at two levels—i.e., sensor level and consumer level—to enable multiple consumers to customize and control virtual sensors corresponding to a single physical sensor.
- Two different multi-tenancy architectures, one for sensor consumers and another for sensor providers. The architecture for sensor consumers offers the highest degree of multi-tenancy to enable sharing of the same application for login, engagement, and management of sensors and sensor data. The architecture for providers provides instances dedicated to each sensor provider, taking into consideration security, failover mechanisms, high availability, and reliability.

- Dynamic provisioning of CSaaS to allow consumers to leverage the vast pool of resources on demand and on a pay-per-use basis.
- Implementation of a SenseCloud prototype system with real sensors, and evaluation to ensure the feasibility of the system and analyze its performance.

The remainder of this paper is organized as follows. Section 2 introduces existing works related to CSaaS or sensor data sharing approaches. Section 3 designs the SenseCloud architecture and explains the system design. Section 4 describes the experiment results for operation and performance. Finally, Section 5 provides concluding remarks with future work.

2. Related Work

Numerous devices around us generate an enormous amount of data. Devices with sensors to capture such data have existed for decades; however, recent developments in technology have enabled these devices to be equipped with energy-efficient and cost-effective wireless modules that allow the devices to transmit data wirelessly in real-time. This feature enables the measurement, inference, and understanding of environmental indicators, ranging from delicate ecologies and natural resources to urban environments. The proliferation of these devices in a communicating-actuating network creates IoT, in which sensors and actuators blend seamlessly with the environment around us, and the information is shared across platforms in order to develop a common operating picture [18]. IoT has been defined as object sensing, object identification, and communication of object-specific information. The information is the sensed data related to temperature, orientation, motion, vibration, acceleration, humidity, chemical changes in the air, etc., depending on the type of sensors. A combination of different sensors can be used for the design of smart services [19].

A multitude of these sensors connected wirelessly form a sensor network. These sensor networks can be leveraged for a variety of applications. Some of these applications, mentioned in [19], are natural-disaster prediction, industrial applications, water-scarcity monitoring, smart-homes design, medical applications, agricultural applications, intelligent transport system design, smart-cities design, smart metering and monitoring, and smart security [12,13,20,21]. However, general users are deterred by the huge investments and high maintenance costs involved in sensor network infrastructure to utilize sensor data. Thus, the concept of CSaaS originated [1,2], and it could be a value-added service to boost the expansion of IoT infrastructure.

The emergence of a plethora of applications for sensor networks is accompanied by several challenges and problems in the management of these networks. The IoT gateway acts as a bridge between WSNs with traditional communication networks or Internet, and it plays an important role in IoT applications; thus, it facilitates the seamless integration of WSNs and mobile communication networks or Internet, and the management and control with WSNs [22]. Some of the problems and challenges that arise are the standardization of governance and management models [3,11,19], complexity arising from heterogeneity [3,4], virtualization, and monitoring [11].

As shown in Table 1, several models have been proposed in order to address these challenges in sensor networks. WSNs as a service [3–5] leverages the widespread use of WSNs and adopts a SOA approach based on the integration of WSNs. In particular, sensing sharing mechanism [5] provides a module to integrate industrial sensor information with the World Wide Web through the cloud in order to monitor and control the development process (e.g., nuclear plant management system). In the system, the integration controller and sensor node communicate through SOA to enable the services to be discovered and invoked by the sensor applications (client). Cloud computing is used to provide the extensibility of application servers and constant availability of data to users. However, the system does not address sensor virtualization, which would enable on-demand sharing of the physical sensors among users.

Service-centric models [6–11] focus on the services provided by a WSN and view a WSN as a service provider. Information as a service (IaaS) for WSNs [9] uses virtualization of WSNs to provide techniques for sensor provisioning and sharing for the large number of existing WSNs.

However, IaaS leaves further scope for expansion in terms of load balancing and universal abstraction for all classes of sensors or sensor networks. A new service model called sensing instrument as a service (SIaaS) [10] provides virtualized sensing instruments to users and shares them as a common resource in a controlled manner. However, SIaaS does not consider on-demand provisioning, which is important to users. Sensor-cloud infrastructure (SCI) [11] manages physical sensors by virtualizing them as virtual sensors on the cloud. SCI provides monitoring, automatic provisioning, and control for virtual sensors and virtual sensor groups, similar to our system; however, SCI does not explicitly design the scalability, load balancing, and multi-tenancy mechanisms to efficiently manage the infrastructure.

Approaches using crowd-sourced mobile devices as sensors have been proposed [12–14]; these approaches utilize the sensors of existing mobile devices and fulfil user requests. A priced public sensing framework (PPSF) [12] is designed for heterogeneous IoT architectures; this framework considers resource limitations in terms of delay, capacity, and lifetime from the perspective of the data provider and considers quality and trust requirements from the perspective of requesters. However, PPSF does not consider the service efficiency of the framework, i.e., scalability or load balancing.

Integrated approaches to share the sensing data on existing sensor networks [15–17] provide efficient sharing mechanisms among multiple applications. A framework of sensor-cloud integration (FSCI) [15] utilizes the ever-expanding sensor data with a content-based pub/sub model. Virtual federated sensor network (VFSN) [16] enables multiple applications to share widely distributed sensor networks flexibly, preserving resource isolations. In VFSN, virtualized sinks are interconnected to achieve a dedicated federated sensor network; further, VFSN provides operations for multiple-sensor information to service providers. An infrastructure for shared sensing (SenseWeb) [17] enables applications to initiate and access sensor data streams from shared sensors across the entire Internet. The SenseWeb infrastructure helps ensure optimal sensor selection for each application and efficient sharing of sensor streams among multiple applications. However, these integration approaches do not address the multi-tenancy challenge that will enable each provider to exclusively and efficiently manage their own resources in a large integrated infrastructure.

In summary, none of the previous studies have developed a prototype of CSaaS with real sensors supporting all the following features: the scaling mechanism, load balancing, virtualization, multi-tenancy, and dynamic provisioning. In this study, we propose and demonstrate SenseCloud, a prototype that implements sensor system management and addresses the above mentioned challenges.

Table 1. Comparison of Sensor Data Sharing Approaches based on Cloud System

	Scale Mechanism	**Load Balancing**	**Sensor Virtualization**	**Explicit Multi-tenancy**	**Dynamic Provisioning**	**System Development**
Sensing Sharing [5]	√				√	√
IaaS [9]	√		√		√	
SIaaS [10]			√			
SCI [11]			√		√	√
PPSF[12]					√	
FSCI [15]			√		√	
VFSN [16]	√	√	√			√
SenseWeb [17]	√	√				√
SenseCloud	√	√	√	√	√	√

3. SenseCloud System

This section describes our proposed SenseCloud system according to a top-down approach: first, the overview and then, the detailed system design.

3.1. Overview of SenseCloud System Architecture

This subsection presents an overview of our SenseCloud system architecture, which consists of three main components—*Entity*, *Cloud Infrastructure*, and *Sensor Network*—as shown in Figure 2.

The four entities are: *Sensor Consumer*, *Sensor Provider*, *Network Admin*, *and Cloud Admin*. The roles of each entity are described below. All entities perform their roles after authentication by the portal server.

- **Sensor Consumer:** First, sensor consumers register on the system and log in. After successful authentication, they subscribe to interesting sensors and create or modify sensor groups that include these sensors. Then, sensor consumers fetch the sensing data from sensors and view the analytical data. Further, they can download historical data and view their bills.
- **Sensor Provider:** Sensor providers register on the system and log in. After successful authentication, they register their sensors on the system, manage the sensors, and control them while checking their status. Sensor providers view the sensor usage statements.
- **Network Admin:** After logging in to the system as a network admin, this entity monitors sensor health and manages virtual sensors. Further, a network admin manages the sensor provider accounts.
- **Cloud Admin:** After logging in to the system as a cloud admin, this entity monitors virtual machines (VMs) and manages cloud infrastructure. A cloud admin manages the sensor consumer accounts and services.

The cloud infrastructure consists of the following servers and storage: *Portal Server*, *SenseCloud Management Server*, *Provisioning Server*, *Sensor Monitoring Server*, *Virtual Server*, *and Data Storage for User/Virtual Sensor Group (VSG)/Sensing Data*. The features of the main components and the roles of each entity are described below:

- **Portal server:** When a user logs in to the SenseCloud portal, the user role, which can be sensor consumer, sensor provider, or admin, determines the operations. In the case of sensor consumers, the dashboard presented by the portal server allows the user to place a request to monitor their virtual sensors, to provision or terminate virtual sensor groups, and to control virtual sensors. In the case of sensor providers, the dashboard provided by the portal server allows the user to register or remove physical sensors. In the case of SenseCloud admins, the dashboard presented by the portal server allows the user to create, modify, and remove the VMs, virtual sensors and virtual sensor groups. The portal server also forwards the requests to other servers when required.
- **Provisioning Server:** The SenseCloud provisioning server creates the virtual sensor groups and manages them according to the requests that are received from the portal server. This task is achieved by the workflow engine and some predefined workflows in the system. The workflow is executed in the following order:

  The provisioning server creates and reserves a VM (if not already created) when it receives the request for provisioning. After the VM is ready, a virtual sensor group is automatically provisioned by the provisioning server. The virtual sensor group is owned by a sensor consumer, and it has one or many virtual sensors. The provisioning server updates the records in the data storage for the virtual sensor groups created by consumers. The consumers can control the virtual sensors. For instance, they can activate or deactivate their subscribed virtual sensors, set the frequency of data, and check the status.
- **Sensor Monitoring Server:** The sensor monitoring server receives the informational or health data about virtual sensors from the virtual servers and stores these data. This information about the virtual sensors is available to sensor consumers on the dashboard. Further, the sensor monitoring server monitors the health status of the physical sensors. This monitoring is

important because live data provisioning is based on the live physical sensors. The admins can also monitor the status of the servers.

- **SenseCloud Management Server:** This server provides the location-aware load balancing algorithm that attempts to select a VM instance that is closest to the request sender zone and that has the shortest pending list. This server provides a multi-tenant solution over the cloud to the registered sensor consumers and providers. It also performs scaling according to the policy engine. This policy engine is based on the network and system performance.
- **Virtual Server:** When requested by the provisioning server, the virtual server creates virtual sensors (VSs) on the VM. The VSs are controlled by the portal server. The virtual server provides health information about the sensors to the sensor monitoring server when requested and saves the information in data storage.
- **Data Storage:** Data storage consists of databases for user, VSG, and sensing data.

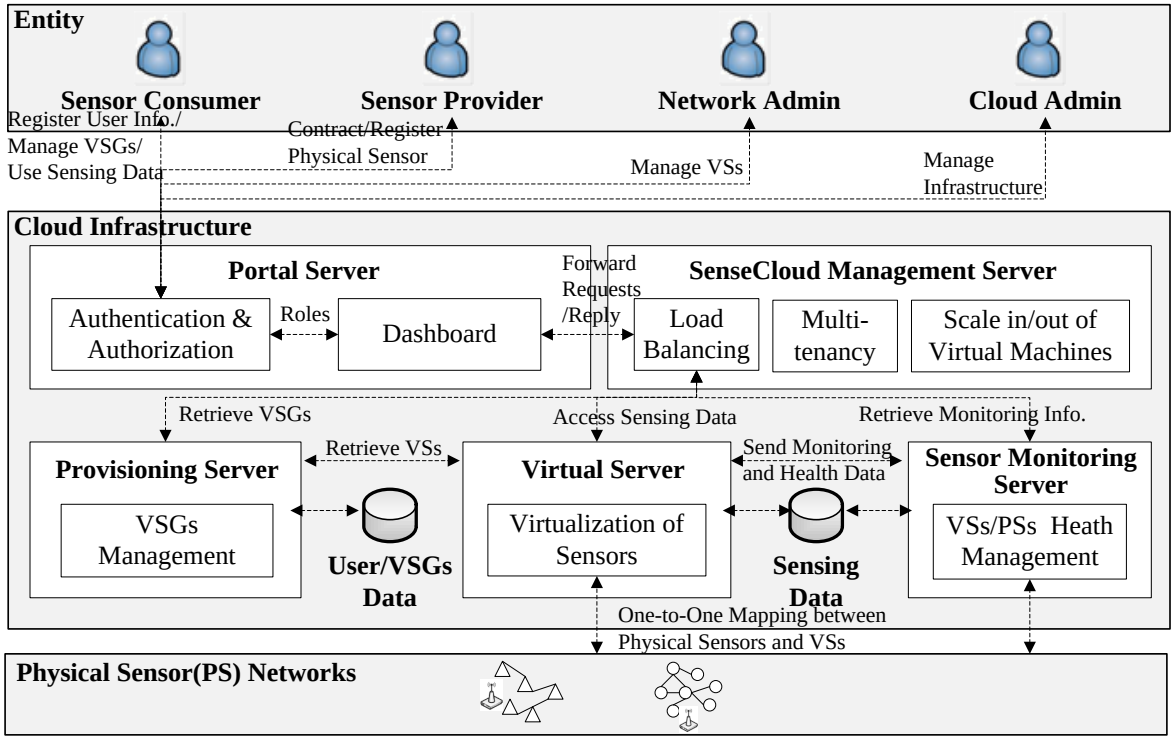


Figure 2. System Architecture of SenseCloud

3.2. System Design

This subsection provides a detailed explanation of the operations of the sensor provider and consumer in SenseCloud by using sequence diagrams, state machine diagram, and active diagram. Then, this subsection presents the functional view of our system.

3.2.1. Operations of Provider and Consumer

Figure 3 depicts the sequence diagram for the workflows of an important entity, i.e., the sensor provider. The sensor provider installs the sensors or sensor network at the corresponding locations. As shown in Figure 3, the sensor provider registers the sensors or sensor network with SenseCloud. After the sensors or sensor networks are registered (registerSensor()), the sensor providers can view them on the dashboard of the portal server, monitor their health (listSensor()), view the usage of their sensors (sensorUsageDetails()), and obtain the monthly usage statement (viewStatement()).

Figure 4 describes the state machine for registerSensor(). When a sensor provider registers for an account, the setup is executed in the cloud. This setup involves the creation of a VM for that sensor provider; the VM will hold all the virtual sensors for every physical sensor plugged in to the cloud by the sensor provider. This setup initiates a process on amazon web service (AWS) using

AWS CloudFormation, creates a VM, and updates the database by recording the assignment of the newly created VM against the entry of the sensor provider. If the setup fails, or if the assignment of the VM to the sensor provider account fails, the process is stopped, and the sensor provider is notified. On de-registering the account, the VM is unassigned, and the user account is removed from the records.

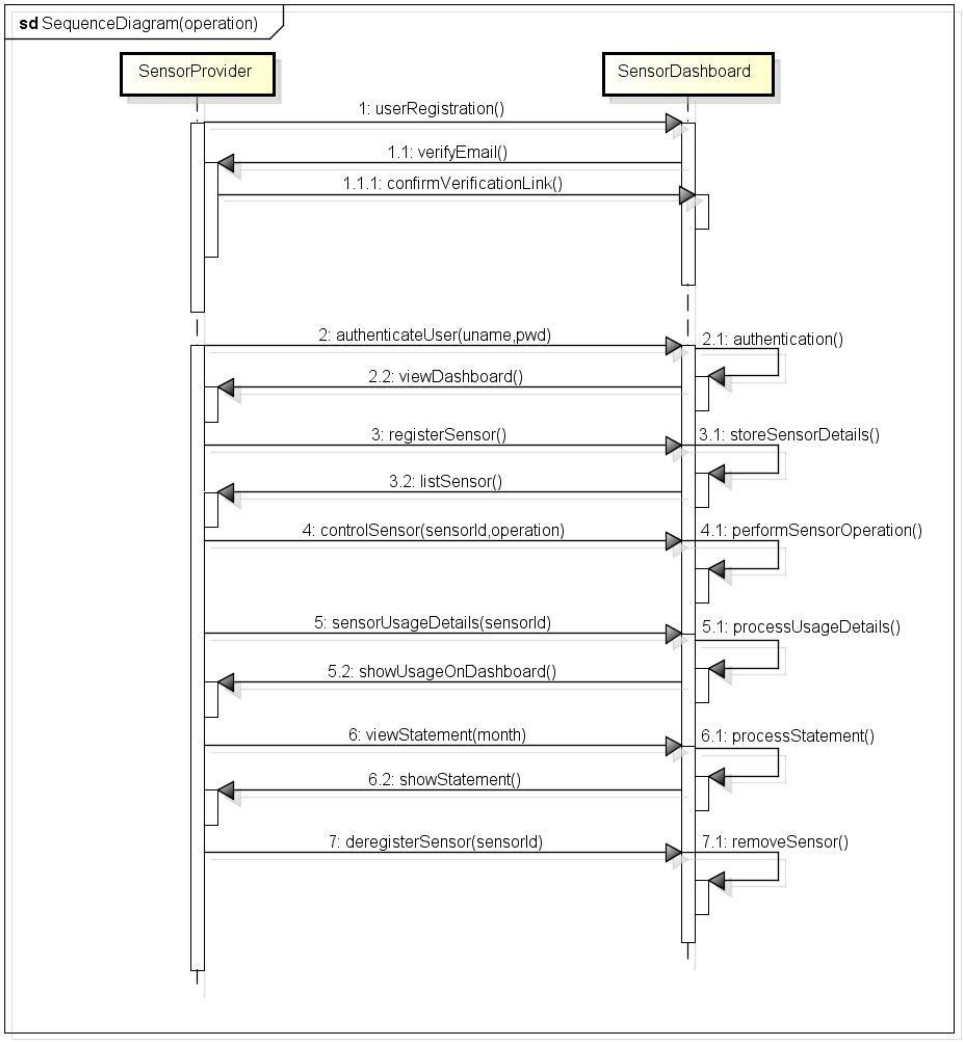


Figure 3. Sequence Diagram for Sensor Provider Workflow

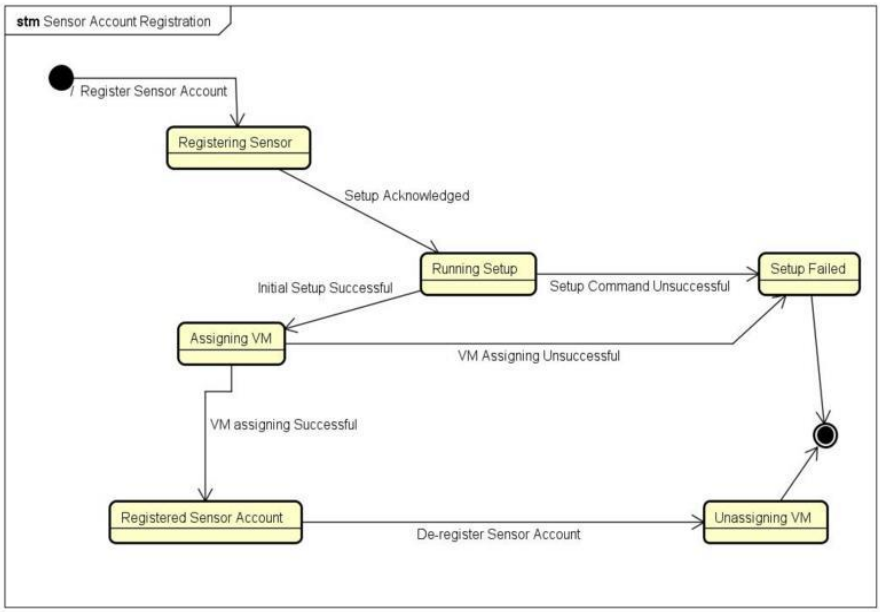


Figure 4. State Machine Diagram for registerSensor() in Figure 3

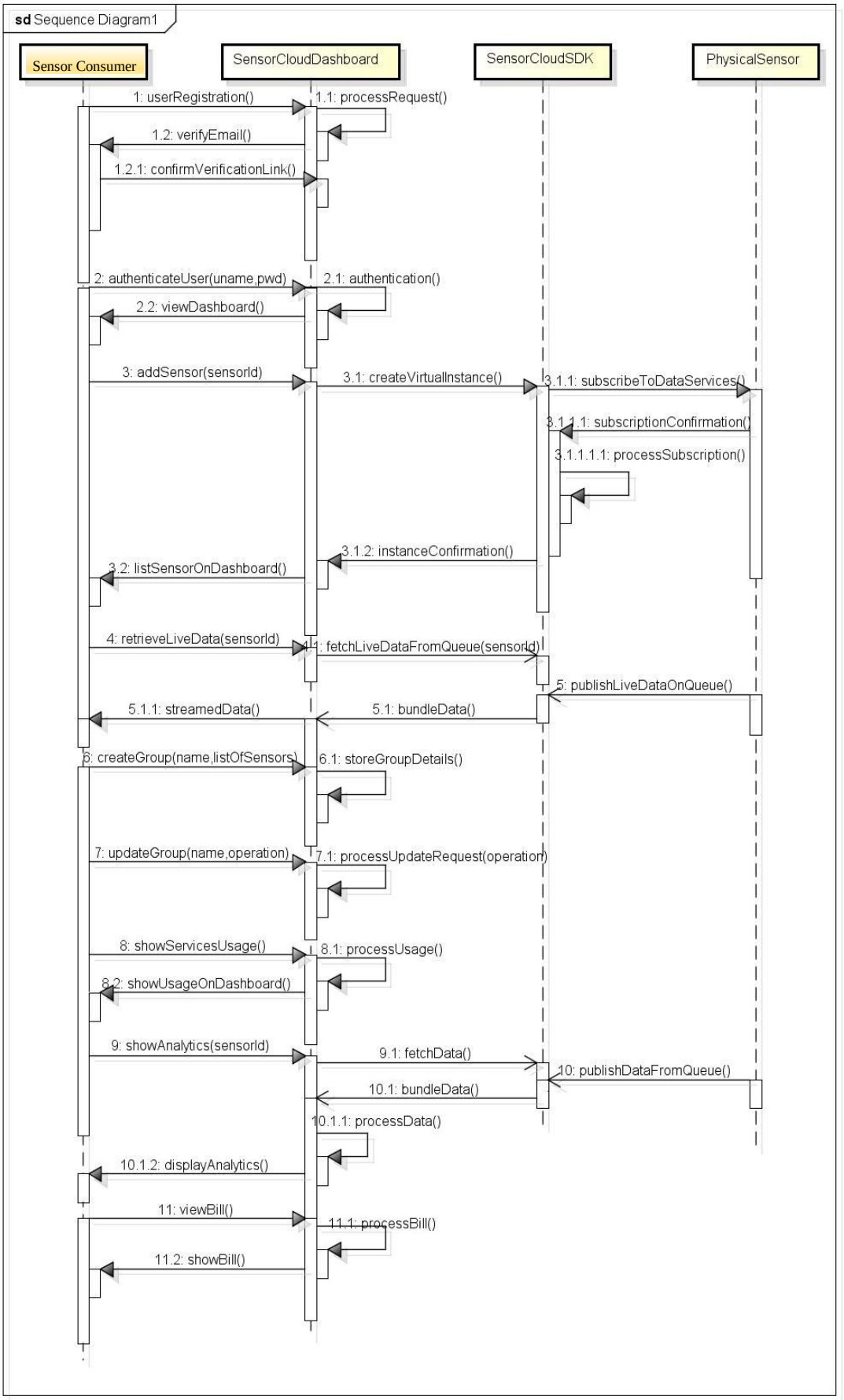


Figure 5. Sequence Diagram for Sensor Consumer Workflow

Figure 5 shows the detailed flow of the sensor consumer operations in SenseCloud. The sensor consumer registers with SenseCloud. After registration, the consumers can view the list of available sensors on their dashboard of the portal server. They can subscribe to any interesting sensors. After subscription, the consumers can club multiple sensors in a group and can manage their groups through the dashboard. The end consumer can view the real-time analytics, download the archived data, and use the developer APIs to utilize the data in their own applications.

Figure 6 illustrates the activity flow of the sensor consumer who subscribes to the sensor, downloads the historical data, and views the analytics on the dashboard. The cloud infrastructure creates a virtual sensor for each physical sensor that the consumer subscribes to. After subscription to a sensor, the consumer can view and download the real-time and historical data in addition to the analytics on the dashboard.

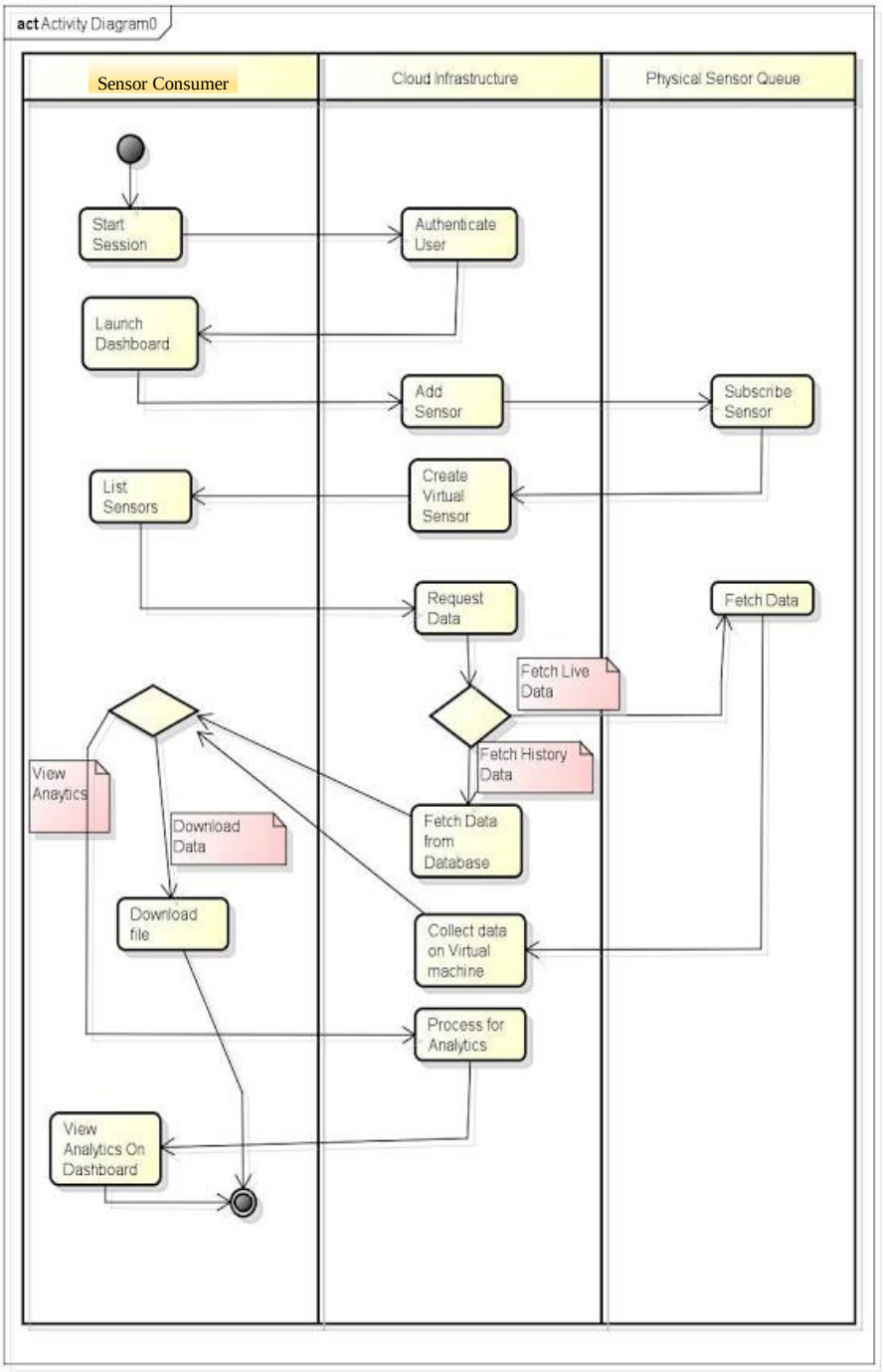


Figure 6. Active Diagram for Sensor Consumer

3.2.2. Functional View of SenseCloud

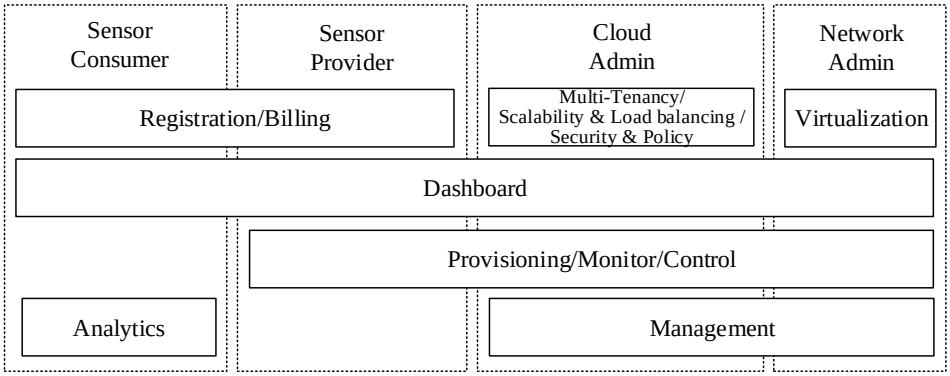


Figure 7. Functions of SenseCloud

Figure 7 shows the functional modules corresponding to each entity in SenseCloud; each module provides the following features:

- **Registration:** The registration module provides consumers with the login and registration capability to enable them to consume the services and data from the sensors via the dashboard. Further, the sensor providers can register their sensors, which will be authenticated by the admins.
- **Virtualization:** The virtualization module virtualizes the sensor network in the cloud by virtually grouping the sensors and services requested by a consumer. Thus, each consumer can group the sensors and services, and can configure, add, edit, and delete the virtualized sensors in the group.
- **Provisioning:** The provisioning module provides a sensor provisioning capability to consumers. The network admin should be able to provision the requested sensor resources to the consumers.
- **Multi-Tenancy:** The multi-tenancy module provides a multi-tenant solution over the cloud to the registered users (i.e., consumers and providers).
- **Scalability & Load Balancing:** The scalability and load balancing module provides easy scalability through an effective and efficient load balancing algorithm. The cloud admin can scale the cloud solution easily. The load balancing ensures the appropriate redirection of user requests to the multiple servers in order to handle the load efficiently.
- **Security & Policy Engine:** The security and policy engine ensures authorization and user-level permissions for different types of users based on configurable policies and security features. The cloud admin can apply and configure policies to restrict or provide different access control based on the user type.
- **Monitor:** The monitor module provides monitoring capability to sensor providers, cloud admin, and network admin. It uses different metrics such as data bitrate, time, and sensor state to gauge bandwidth, performance, health, and billing. The sensor provider can monitor the actual physical state of the sensors to perform maintenance, provisioning, and de-provisioning of sensors. The cloud admin can monitor the VMs. The network admin can monitor the resource usage for billing and the sensor state for health.
- **Control:** Similar to the monitor module, the control module provides controlling capability to the sensor provider and admins in order to control the users and sensor network services.
- **Billing:** The billing module provides billing to the consumer according to the usage of data and services. The billing is based on a configurable cost model. The sensor providers can view the usage of their sensors and obtain the monthly usage statement.
- **Analytics:** Sensor consumers can access real-time sensor data analysis and archived sensor data analysis on the dashboard.
- **Management:** The management module enables the network admin to manage the virtual sensors. The network admin can create virtual groups with virtual sensors. Further, this module

provides user account management and cloud infrastructure management capabilities to the cloud admin.

We focus on the design of virtualization, multi-tenancy, and dynamic provisioning. We explain our solutions that address these aspects.

1) Virtualization Solution

In order to obtain the typical benefits of virtualization and provide an abstraction and isolation to the consumers, the virtualization in SenseCloud is composed of two levels (similar to the model in [11]), as shown in Figure 8; these levels are: *Sensor-Level Virtualization* and *Consumer-Level Virtualization*. *Sensor-Level Virtualization* refers to the actual virtualization of physical sensors to VMs. *Consumer-Level Virtualization* refers to the logical grouping of virtual sensors, thus providing an abstraction and isolation to the consumers.

In *Sensor-Level Virtualization*, SenseCloud virtualizes physical sensors from providers to virtualized instances available to individual consumers. Besides sensor virtualization, we allocate a VM instance to each sensor provider to register, manage, and monitor their sensors and sensor networks, as shown in Figures 8 and 9. In *Consumer-Level Virtualization*, if a consumer subscribes to any of the sensors, the sensors are virtualized, and multiple virtualized sensors are grouped together. This process allows multiple consumers to customize and control virtual sensors corresponding to a single physical sensor. For example, a consumer *C1* could group the temperature sensor (virtual sensor) and light sensor (virtual sensor) as virtual sensor group *VSG1*, and another consumer *C2* could group the temperature sensor (virtual sensor) and pressure sensor (virtual sensor) as virtual sensor group *VSG2*. These two virtual groups (*VSG1* and *VSG2*) are independent of each other and can be customized by the consumer.

Although our system consists of two levels of virtualization, the sensor data are not replicated by each virtual sensor, and the data from each sensor are stored in a common distributed database. Thus, instead of replication of data, the data are shared with consumers who subscribe to the sensors and own the virtual sensors in their virtual sensor group.

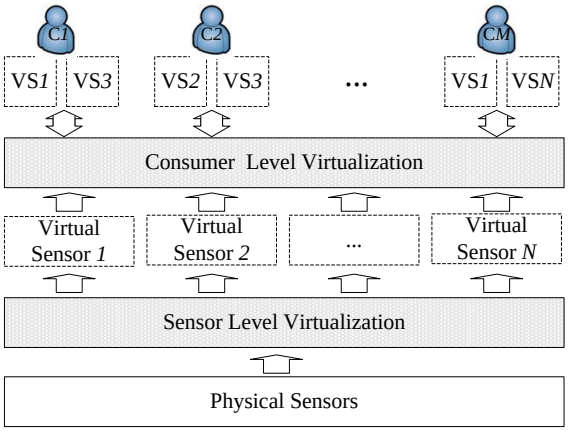


Figure 8. SenseCloud Virtualization Approach (C*i* is the *i*th sensor consumer, and VS*j* is the *j*th virtual sensor.)

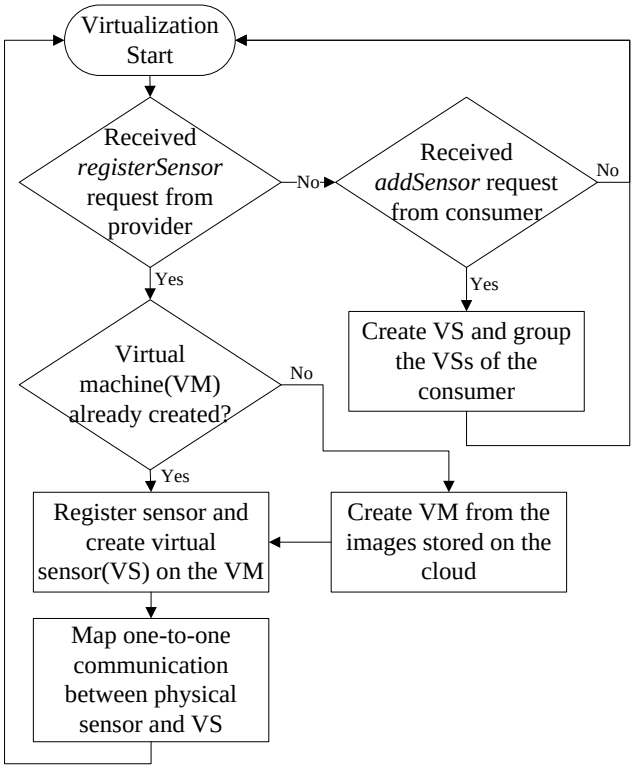


Figure 9. SenseCloud Virtualization Algorithm

2) Multi-tenancy Solution

As shown in Figure 10, SenseCloud has two different perspectives for multi-tenancy: perspective of sensor consumer and perspective of sensor provider.

Let us consider SenseCloud from the perspective of the sensor consumer. SenseCloud has the highest degree of multi-tenancy from this perspective. Sensor consumers share the same application to log in, engage, and manage sensors and sensor data. The application servers (i.e., provisioning servers in the infrastructure) are also shared among various sensor consumers. These application server instances can scale out or in according to the policy engine (refer to 3) Dynamic Provisioning below). The data corresponding to the sensor consumers are stored in shared tables in a common database (i.e., MySQL).

Next, let us consider SenseCloud from the perspective of the sensor provider. Each sensor provider has dedicated infrastructure (i.e., VMs). Each of the instances dedicated to the sensor provider runs its own software stack exclusively for the particular sensor provider. This design considers the following important aspects: security, failover mechanisms, high availability, and reliability. For each sensor provider, several sensors can post data to their specific instances. In case of data corruption, failover, or deliberate attacks, the specific sensor provider can be isolated without affecting or compromising the entire application. However, the data collected from the sensors across sensor providers are stored in a shared database (i.e., Cassandra DB) after filtering and validation. This step is performed to enable analytics over the entire data set.

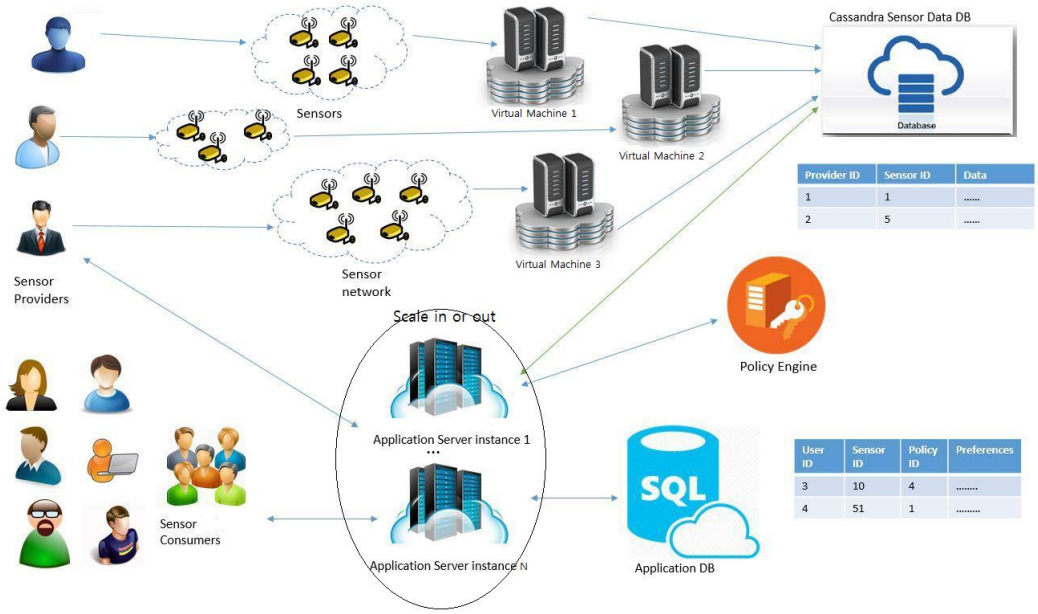


Figure 10. SenseCloud Multi-Tenancy Solution

Further, the multi-tenancy solution in SenseCloud spans three levels: sensor level, sensor-data level, and sensor-service level. In the multi-tenancy view of the sensor level, SenseCloud provides diverse sensors and sensor networks to each tenant—i.e., consumer—according to the individual demand. Our cloud infrastructure of sensors provides customized composition, provision, and schedule of sensors for each tenant. SenseCloud provides sensor-data level multi-tenancy—i.e., customized data collection, data management, and data visualization—according to the demand of each tenant. Further, our system provides sensor-service level multi-tenancy, e.g., real-time data service and historical data service. In addition, this feature could be enhanced to provide prediction and recommendation services through in-depth analysis of historical data.

3) Dynamic Provisioning

In order to allow the sensor consumers to efficiently leverage the vast pool of resources on demand and on a pay-per-use basis, we provide two solutions: scalability solution and load balancing solution.

Our scaling solution considers three parameters for scaling in and out. We consider the average incoming network traffic per instance, average outgoing network traffic per instance, and average CPU utilization per instance. As shown in Figure 11, we dynamically set thresholds for each of these parameters according to the usage patterns. When surpassed, these thresholds trigger the dynamic creation and deletion of instances to efficiently scale out and scale in, respectively.

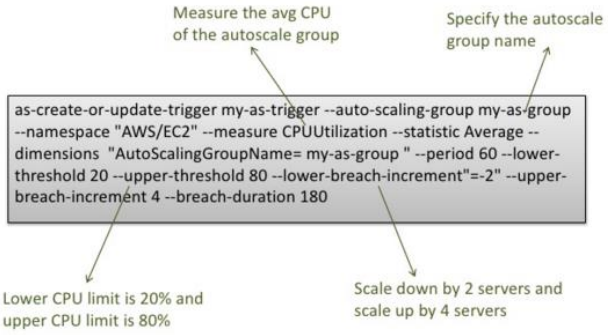


Figure 11. Example of SenseCloud Scaling Policy

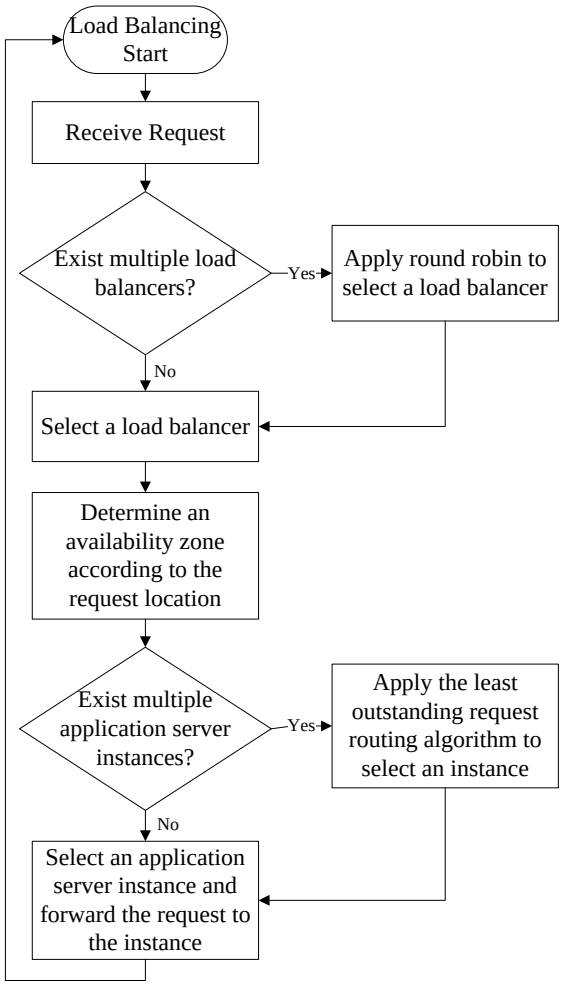


Figure 12. SenseCloud Load Balancing Algorithm

The load balancing solution is shown in Figure 12. When an application server (e.g., provisioning server) receives the request, it selects a load balancer from the multiple load balancers by using the round robin algorithm. If only one load balancer exists, it is selected by default. The selected load balancer uses the location-aware load balancing algorithm in conjunction with the least outstanding request routing algorithm. Thus, the location-aware load balancing algorithm attempts to select an instance that is closest to the request sender zone and that has the shortest pending list. First, the load balancer selects the availability zone closest to the location where the request is sent. Then, the load balancer determines the application server instances that exist in the selected availability zone. If multiple application server instances exist, the load balancer selects an instance according to the least outstanding request routing algorithm, i.e., it selects an instance that has the smallest queue size for outstanding requests.

4. Performance Evaluation

This section describes the implementation of a prototype of SenseCloud, thus demonstrating the feasibility of our system. Then, the prototype is evaluated in terms of the system response time and the performance of our functional algorithms.

4.1. Evaluation of Implementation

Table 2 shows the tools and technologies that we used to implement a prototype of SenseCloud and to test it.

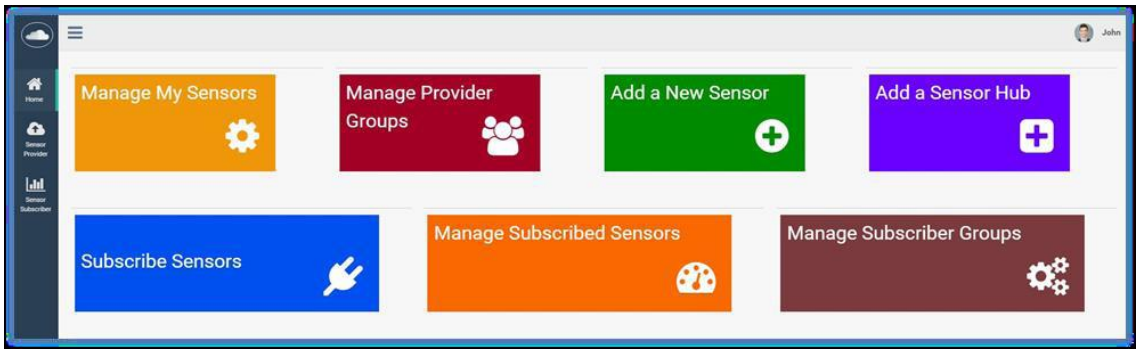
Table 2. Tools and Technologies

Tools and Technologies	Name	Version	Description
Development	Java, Python, Groovy, JS	NA	Independent programming language.
API Framework	Swagger	1.5	Representation of RESTful API
Operating System	Linux	14	Open-source operating system
White-box Testing Framework	JUnit	3.8.1	Dedicated testing framework for Java with native support
Black-box Testing Framework	Selenium	2	Automated testing framework for web applications
Integrated Development Environment	Eclipse	Luna	Widely used IDE with full support for Java and third party plugins
Communication Protocol	MQTT	2.3.1	Lightweight publish-subscribe broker communication protocol for IoT
Storage	Cassandra	2.1.7	Big data storage on clustered commodity hardware
	MySQL	5.1.36	SQL database to store dashboard-related information
Continuous Integration Framework	Jenkins	1	Continuous integration for software development. Also supports Git repository
UI	HTML	5	Fundamental and flexible Web UI language for web application
	CSS	3	Fundamental and flexible Web UI styling for web application
Web	JQuery	2	DOM manipulation library
	moris.js	0.5.1	Graphical data visualization Java Script library
	Bootstrap	3	Java Script framework to build responsive websites
Repository	Git	1.9	Open-source sub version repository. Powerful collaboration, management, and code review for projects
Cloud	AWS	NA	Amazon Web Services to build cloud infrastructure
Build Tool	Maven	3.2.1	Build automation system to automate build, testing, publishing, and deployment activities
UML	Astah	6	Design UML diagrams
MQTT Client	Eclipse Paho	0.4.0	Open-source client implementation of MQTT
Logging	Log4j	1.2.17	Library for logging in Java
Sensor Kit	RasberryPi, Samsung SmartThings	NA	10 micro-controllers with sensors: RasberryPi has temperature sensor, pressure sensor, ambient light sensor, and LED; SmartThings has temperature sensor, contact sensor, humidity sensor, motion sensor, and orientation sensor

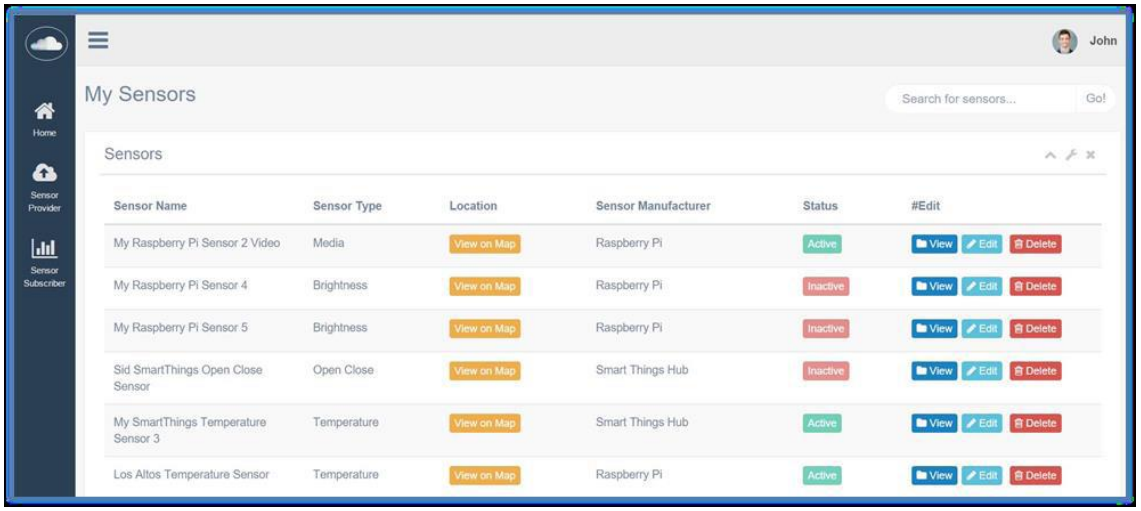
We present some screenshots of our implemented prototype. Users type the SenseCloud index page URL in the browser. After successful registration, the users can log in to their account by entering their credentials. After successful login, the users are redirected to the main dashboard page, which shows the links to the functions that the users can perform, as shown in Figure 13(a).

Sensor providers can click the "Manage My Sensors" menu to view, edit, and delete the sensors that they have registered; Figure 13(b) shows the sensor list of a sensor provider. Sensor providers can add sensors by clicking the "Add a Sensor" menu. They can group their sensors and manage the groups through the "Manage Provider Groups" menu; further, they can add sensor hubs by selecting the "Add a Sensor Hub" menu in order to create a sensor network.

Sensor consumers can subscribe to interesting sensors through the "Subscribe Sensors" menu, view the list of available sensors, search for sensors, and subscribe to these sensors. Sensor consumers can manage their subscriptions and view the data of the subscribed sensors through the "Manage Subscribed Sensors" menu. The subscribed sensors can also be grouped together; then, the consumer can view, edit, manage, and visualize the data from the sensors of the created group by selecting the "Manage Subscriber Group" menu. Figure 13(c) shows the temperature sensor data for a subscribed sensor. The consumer can view the daily current, minimum, maximum, and average temperature and can also download the historical data in JSON format by selecting the date. Figure 13(d) shows the maximum and minimum temperatures of the current week for a subscribed sensor. A map displays the current geo-location of the sensor. The implementation of the SenseCloud prototype confirms the feasibility of our system.

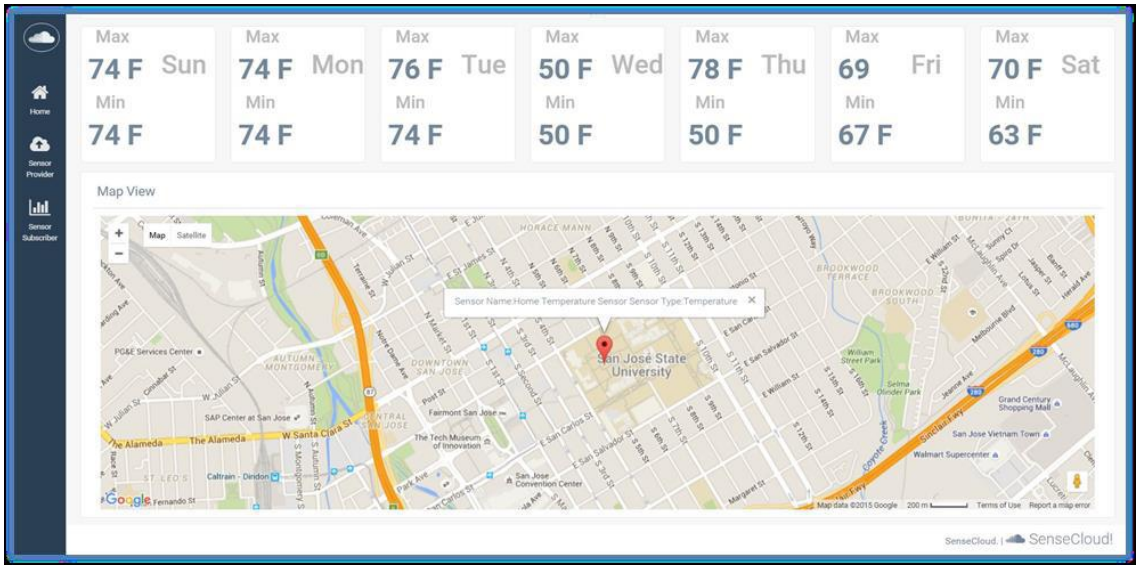


(**a**)



(b)

(c)



(d)

Figure 13. Screenshots of SenseCloud Prototypes: **(a)** Menu Page; **(b)** Sensors Viewed through "Manage My Sensors" Menu; **(c)** Sensor Historical Data through "View" Menu of Each Sensor; **(d)** View Sensor Data with Map through "View on Map" Menu of each Subscribed Sensor

4.2 Performance Observation

In order to evaluate the performance of our prototype, we develop the servers of SenseCloud on AWS cloud infrastructure and generate client requests to the servers. We observe the system response time, varying the request types of each scenario, the number of requests, and applied performance-tuning mechanisms. Many requests more than 1000 is the statistically meaningful. Finally, we measure request distributions to show the performance of our functional algorithms (i.e., load balancing and scalability).

First, in order to evaluate the system response time, we create three scenarios by varying the number of requests to list the subscribed sensor and the number of requests to add sensors. Further, in a scenario, we increase the number of users from 1 to 100. The requests from a single user correspond to synchronous calls, and different users create simultaneous connections

As shown in Tables 3 and 4, the performance remains the same even if we increase the number of synchronous requests from 1000 to 5000. This result is an effect of the multi-tenant architecture

and load balancer algorithm running on the VMs; owing to this design, the incoming requests are handled in parallel, and the same throughput is obtained. Further, when the number of simultaneous connections increases, we observe that even if the number of users increases, the performance deteriorates only by a small amount.

Table 3. System Response Time vs. Requests to List the Subscribed Sensors

Scenario	Number of Requests			Average (Deviation) Response Time (ms)
	Requests to List the Subscribed Sensors per User	Users	Total Requests	
1	1000	1	1000	6 (0)
2	5000	1	5000	6 (7)
3	1000	100	100000	27 (63)

Table 4. System Response Time vs. Requests to Add Sensors

Scenario	Number of Requests			Average (Deviation) Response Time (ms)
	Requests to Add Sensors per User	Users	Total Requests	
1	1000	1	1000	11 (2)
2	5000	1	5000	11 (0)
3	1000	100	100000	1177 (78)

Next, we create eight scenarios, varying the number of requests to perform the following 13 page traverses and applying mechanisms for performance tuning (i.e., connection pooling, prepared statement, and object caching). A request includes traversals of the following 13 pages:

1. Home page
2. Registration
3. Login
4. Add sensor to cloud
5. List provided sensors
6. Group of provided sensors
7. Consumer dashboard
8. Subscribe sensors
9. My Sensors
10. My Groups
11. Dashboard for temperature sensor analysis
12. Dashboard for temperature and video sensor analysis in my group
13. Logout

From the results in Table 5, the use of other performance-tuning mechanisms such as connection pooling, prepared statement, and object caching, has significantly reduced the average time for a request response. Our system implements connection pooling by maintaining a connection pool of database connections and using the queue mechanism to dequeue or enqueue database connection objects. This process improves the performance significantly because the connection objects are reused. Instead of traditional statements, prepared statements are used for database querying. The prepared statements are pre-compiled. Object caching is also implemented in our system; the most frequently queried results are cached based on the time-to-live.

Table 5. System Response Time vs. Requests & Applied Mechanisms

Scenario	Number of Requests			Applied mechanisms			Average (Deviation) Response Time (ms)
	Requests per User	Users	Total Requests	Connection Pooling	Prepared Statement	Caching	
1	1000	1	1000				65 (86)
2	1000	1	1000	√			54 (73)
3	1000	1	1000	√	√		42 (66)
4	1000	1	1000	√	√	√	13 (17)
5	1000	10	10000				401 (392)
6	1000	10	10000	√			307 (502)
7	1000	10	10000	√	√		250 (417)
8	1000	10	10000	√	√	√	125 (84)

In order to evaluate the performance of our load balancing and scalability algorithms, we use a request generator that, at a given time, randomly generates 1000 requests across nine locations. Initially, each location has one instance. All instances have different configurations created in the Elastic Cloud Compute service of Amazon Web Services (AWS). The nine locations on AWS are as follows [23]:

1. US East (N. Virginia)
2. US West (Oregon)
3. US West (N. California)
4. EU (Ireland)
5. Asia Pacific (Singapore)
6. Asia Pacific (Tokyo)
7. Asia Pacific (Sydney)
8. Asia Pacific (Seoul)
9. South America (Sao Paulo)

Figure 14 is a graphical representation of the request distribution results according to the applied algorithms. From the results shown in Figure 12(a), in the absence of the algorithms, all the 1000 requests were randomly distributed across all the locations. Thus, the result of request distribution is the most uneven in this case. In Figure 12(b), only the load balancing algorithm is applied; the load balancer attempts to distribute the requests originating from a particular location until the capacity of the server to serve requests reaches a maximum. If overload occurs, the requests are redirected to different locations randomly, using the least pending request algorithm. For example, location 1 received 121 requests, and according to the load balancing algorithm, the first 101 requests are handled by the instance at location 1. After the instance at location 1 reaches its threshold, the subsequent requests are directed to other locations by using the least outstanding request routing algorithm. In Figure 12(c), only the scalability algorithm is enabled. When a particular location experiences a surge in network traffic and high CPU usage that reaches the threshold values, our system scales the infrastructure for that location. For example, when the

instance at location 1 reached its threshold, the infrastructure at location 1 scaled out, and the subsequent requests were handled by these instances at location 1. In Figure 12(d), both algorithms are enabled, and they work in conjunction to serve the requests. For example, location 1 received 202 requests. Based on the load balancing algorithm and these requests, location 1 scaled out to serve these requests. A similar trend is observed for the other locations.

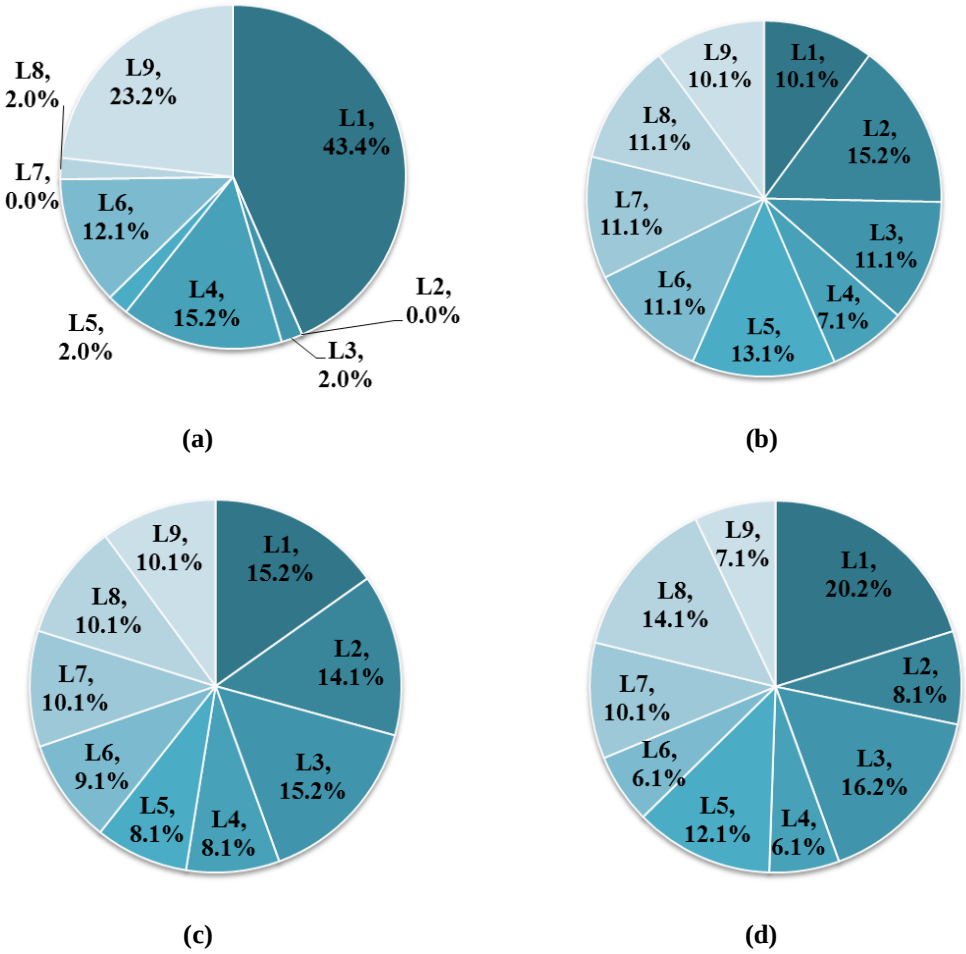


Figure 14. Request Distributions with/without Our Load Balancing and Scalability Algorithms on AWS (L*i* denotes location *i*): **(a)** Without Algorithms; **(b)** With Load Balancing; **(c)** With Scalability Algorithm; **(d)** With Load Balancing and Scalability Algorithms

5. Conclusions and Future Work

SenseCloud is a cloud platform that addresses the challenges of virtualization, multi-tenancy, and dynamic provisioning encountered by the IoT industry today. Our cloud infrastructure provides a layer that connects with different sensor networks, resolves the connectivity and engagement concerns, and efficiently provides CSaaS between sensor providers and consumers. SenseCloud provides a two-level virtualization mechanism. It virtualizes the physical sensors to enable the consumers to utilize them without apprehensions about the specification and location details. Further, it allows the consumers to place dynamic requests for virtual sensor groups and to customize the virtual sensor groups. SenseCloud provides different types of multi-tenancy to sensor providers and consumers: It provides a virtual instance dedicated to each provider to enable isolation without affecting the other providers in case of failover or attacks; further, it also provides common application server instances to all consumers for efficient sharing of resources. SenseCloud

provides dynamic provisioning to allow the consumers to leverage the vast pool of resources on demand and on a pay-per-use basis. The results from our prototype implementation and simulation have demonstrated the feasibility of SenseCloud and the achievement of our objectives.

In future work, we must consider the large number of IoT devices and the exponential growth of sensor manufacturers in the market. Instead of adopting a de-facto standard, we must define a specific standard for communication, data format, and security between the devices and the cloud platform. The definition of such a standard will eliminate the necessity of frequent changes in the cloud implementation when a new manufacturer enters the market. Current security in SenseCloud is imposed by the role-based access control and provision policy and the two levels of virtualization. With advanced and proper security being the most important feature in the world of connected devices, further development and advancement is required to increase the trust in such cloud platforms among consumers.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grant number 2015R1D1A1A01057362).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Botta, A.; Donato, W.; Persico, V.; Pescape, A. On the Integration of Cloud Computing and Internet of Things. In Proceedings of the International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 27–29 Aug. 2014; pp. 23–30.
2. Sheng, X.; Tang, J.; Xiao, X.; Xue, G. Sensing as a Service: Challenges, Solutions and Future Directions. *IEEE Sensors Journal* **2013**, 13, 3733-3741.
3. Delicato, F.C.; Pires, P. F.; Pirmez, L. Wireless Sensor Networks as a Service. In Proceedings of 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS), 2010, Oxford, England, 22-26 Mar. 2010; pp. 410–417.
4. Zhu, Q.; Wang, R.; Chen, Q.; Liu, Y.; Qin, W. IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things. In Proceedings of 8th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC), Hong Kong, 11–13 Dec. 2010, pp. 347–352.
5. Rajesh, V.; Pandithurai, O.; Mageshkumar, S. Wireless Sensor Node Data on Cloud. In Proceedings of IEEE International Conference on Communication Control and Computing Technologies (ICCCCT), Tamil Nadu, India, 7–9 Oct. 2010, pp. 476–481.
6. Zaslavsky, A.; Perera, C.; Georgakopoulos, D. Sensing as a Service and Big Data. In Proceedings of International Conference on Advances in Cloud Computing (ACC), Bangalore, India, 26–28 Jul. 2012, pp. 21–29.
7. Gracanin, D.; Eltoweissy, M.; Wadaa, A. A Service-Centric Model for Wireless Sensor Networks. *IEEE J. Sel. Areas Commun*. **2005**, 23, pp. 1159–1166, DOI: 10.1109/JSAC.2005.845625.
8. Distefano, S.; Merlino, G.; Puliafito, A. Sensing and Actuation as a Service: A New Development for Clouds. In Proceedings of 11th IEEE International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 23–25 Aug, 2012. pp. 272–275.
9. Deshwal, A.; Kohli, S.; Chethan, K.P. Information as a Service Based Architectural Solution for WSN. In Proceedings of 1st IEEE International Conference on Communications in China (ICCC), Beijing, China, 15–17 Aug. 2012, pp. 68–73.
10. Lauro, R. D.; Lucarelli, F.; Montella, R. SIaaS - Sensing Instrument as a Service Using Cloud Computing to Turn Physical Instrument into Ubiquitous Service. In Proceedings of IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), Leganes, 10–13 Jul, 2012. pp. 861–862.

11. Al-Fagih, M.A.E.; Al-Turjman, F.M.; Alsalih, W.M.; Hassanein, H. S. A Priced Public Sensing Framework for Heterogeneous IoT Architectures. *IEEE Trans. Emerg. Topics Comput.* **2013**, 1, pp. 133–147, DOI: 10.1109/TETC.2013.2278698J.
12. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Sensing as a service model for smart cities supported by Internet of Things. *T. Emerg. Telecommun. T.* **2014**, pp. 81–93, DOI: 10.1002/ett.2704.
13. Sheng, X.; Tang, J.; Xiao, X. Sensing as a Service: Challenges, Solutions and Future Directions. *IEEE Sensors J.* **2013**, 13, pp. 3733–3741, DOI: 10.1109/JSEN.2013.2262677.
14. Hassan, M.M.; Song, B.; Huh, E. A Framework of Sensor - Cloud Integration Opportunities and Challenges. In Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC), Suwon, Republic of Korea, 15–16 Jan. 2009, pp. 618–626.
15. Ishi, Y.; Kawakami, T.; Yoshihisa, T.; Teranishi, Y.; Nakauchi K.; Nishinaga, N. Design and Implementation of Sensor Data Sharing Platform for Virtualized Wide Area Sensor Networks. In Proceedings of seventh IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Victoria, BC, 12–14 Nov. 2012, pp. 333–338.
16. Kansal, A.; Nath, S.; Liu, J.; Zhao, F. SenseWeb: An Infrastructure for Shared Sensing. *IEEE Multimedia Mag.* **2007**, 14, pp. 8–13, DOI: 10.1109/MMUL.2007.82.
17. Gubbi, J.; Buyya, R.; Murasic, S.; Palaniswami, M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Gener. Comp. Sy.* **2013**, 29, pp. 1645–1660, DOI: 10.1016/j.future.2013.01.010.
18. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In Proceedings of 10th IEEE International Conference on Frontiers of Information Technology (FIT) Islamabad, Pakistan, 17–19 Dec. 2012. pp. 257–260.
19. Hoang, D.B.; Chen, L. Mobile Cloud for Assistive Healthcare. In Proceedings of IEEE Asia-Pacifi Services Computing Conference (APSCC), Hangzhou, China, 6–10 Dec. 2010. pp. 325–332.
20. Rao, B.B.P.; Saluja, P.; Sharma, N. Cloud Computing for Internet of Things & Sensing based Applications. In Proceedings of Sixth IEEE International Conference on Sensing Technology (ICST), Kolkata, 18–21 Dec. 2012, pp. 374–380.
21. Yuriyama, M.; Kushida, T. Sensor-Cloud Infrastructure- Physical Sensor Management with Virtualized Sensors on Cloud Computing. In Proceedings of 13th IEEE International Conference on Network-Based Information Systems (NBiS), Takayama, Gifu, Japan, 14–16 Sep. 2010. pp. 1–8.
22. Ibbotson, J.; Gibson, C.; Wright, J. Sensors as a Service Oriented Architecture: Middleware for Sensor Networks. In Proceedings of Sixth IEEE International Conference on Intelligent Environments (IE), Kuala Lumpur, Malaysia, 19–20 Jul. 2010. pp. 209–214.
23. Regions and Availability Zones. Available online: URL: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-regions-availability-zones (accessed on 1 Dec. 2015).