Ain Shams University, Faculty of Engineering

Ain Shams Journal of Electrical Engineering (ASJEE)

Special Issue January 2017

Service Level Agreement in Sensor-Cloud

Ghazaleh Ayoubi*, Saeed Sedighian Kashi, Nazanin Mashhadi Tafreshi
Department of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran

Abstract

Internet of Things (IoT) is one of the topics that taken into consideration in recent years. Sensor-Cloud is an interested topic for IoT researchers. In Sensor-Cloud, owners of the sensor network can transmit their data to the cloud providers, so users or applications can request their demands from cloud providers without worry about low-level details like the location and the type of sensor nodes. In Sensor-Cloud, user satisfaction must be addressed in a user-friendly manner. In this paper, we propose a four-layered Sensor-Cloud architecture that contains the sensor network layer, the cloud provider layer, the broker layer, and the application layer. The sensor network owners and the cloud providers have different Service Level Agreement (SLA) parameters. The sensor network owners can register in one or more cloud providers and then give their SLA parameters to them. Also, the cloud providers are connected to one or more brokers. Each of brokers tries to find the best provider and introduce it to the user with respect to the user requirements. Brokers use one of the methods of Multi-Attribute Decision Making (MADM) to find the best cloud provider, namely Simple Additive Weighting (SAW). When a user receives offers of brokers, by reusing of SAW method chooses the best cloud provider among offers. In this paper functionality of SAW method in several different configurations of four-layered architecture is studied. The simulation results show that the applications that are connected to more brokers can receive better offers.

1. Introduction

In recent years especially from 2012 onwards, Internet of Things (IoT) is the center of attention in information technology (IT). In the IoT, many of devices even those we use them in daily life will be on the network or connect to the internet. Via the interconnection of sensing and actuating devices, collecting and sharing of information becomes possible [1]. The interconnection between devices especially via the wireless communications is already possible in Wireless Sensor Networks (WSNs).

According to [2]–[4], a WSN is a self-organizing network of distributed devices or nodes, which are spatially distributed and work cooperatively to sense the environment, achieve a common goal, exchange gathered information through wireless links and also control the environment. With the emergence of the IoT in recent years, use of WSNs is on the rise, so data that comes from WSNs is growing explosively. Because the amount of this data is enormous (high-volume), and because this data is begin generated fast (high-velocity) and is in many forms (high-variety), and due to the limitation of WSNs; manage, process, analysis and maintenance of this information by the WSNs is not possible and requires an appropriate framework. Cloud computing is very suitable for this purpose because, with the use of it, we can store and process big data.

One of the popular topics that emerge in IoT and is taken into consideration is Sensor-Cloud. The authors of [3] define Sensor-Cloud as a heterogeneous computing environment that brings together multiple WSNs consisting of different sensors. The authors of [5] believe that Sensor-Cloud infrastructure virtualizes physical sensor nodes. Sensor-Cloud provides sensing as a service, sensor event as a service, and sensor data as a service. Indeed Sensor-Cloud consist of WSNs and cloud computing resources, in this environment applications can provision and release their requirements, in an automatic and dynamic manner [3], [5]–[7]. Sensor-Cloud brings some advantages for applications and users of it. Some of these advantages are 1) Removing the low-level details: with the use of Sensor-Cloud low-level details like configuration details of sensors, the location of them, etc. will be eliminated [3], [5], [8], 2) Sharing sensor data: by using the virtual sensors, sharing sensor data among multiple users will be possible, so the

* Corresponding Author: ghazaleh_ayoubi@mail.kntu.ac.ir

cost of collecting information reduce for both of system and user and since reusability of data is also transparent to the user, receiving redundant data is reduced and as a result, the performance increases [3], and 3) Better management of sensors: the integration of sensors into the cloud enables users to easily collect, access, search, process, analysis and share large amount of sensor data [6]–[8]. In fact with the use of Sensor-Cloud, users can benefit from the various advantages of the cloud computing including scalability, virtualization, measured services, etc.

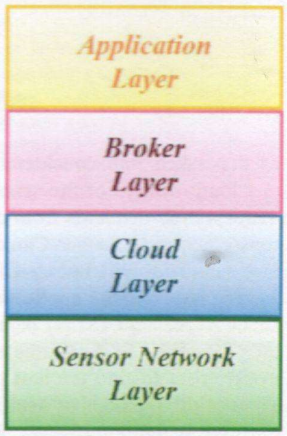

Figure1. Four-layer architecture of Sensor-Cloud

As seen in Figure1, we propose a four-layered Sensor-Cloud architecture that contains the sensor network layer, the cloud provider layer, the broker layer, and the application layer. Considering there are many cloud providers and WSN providers in Sensor-Cloud, applications and users would like to get the best service level agreement (SLA) from the Sensor-Cloud framework, without involved in the details. For this reason, we assume some brokers in the layer 3 of our four-layered architecture take the responsibility to deliver application requirements with the best possible SLA.

Each of brokers is connected to one or more cloud providers and applications. A broker consists of receiver component and calculator component. The receiver component receives SLA data from cloud providers that are connected to it, on the other side it receives requirements of applications that are connected to it. Then broker's calculator component with the use of Multi-Attribute Decision Making (MADM) method finds a best possible cloud provider that can meet the SLA requirements of the application. To reach this goal, in this work we implement one of the most widely used methods of MADM in brokers, namely Simple Additive Weighting (SAW) [9], [10]. The application receives one alternative from each of brokers that are connected to it and then select the best one. This paper is organized as follows. Section 2 is related work. In Section 3 we illustrate the four-layered architecture of Sensor-Cloud and also SAW method that is implemented in the broker's calculator component. Section 4 and 5 include analysis and conclusion respectively.

2. Related Work

In different resources, authors use different approaches for integrating WSN and cloud. In [3] the authors believe that Sensor-Cloud decouples sensor owner and users, they say that sensor cloud is composed of virtual sensors, built on top of the physical sensors. In this paper, virtual sensor is an emulation of the physical sensor. Authors of [3] implement virtual sensor in four configurations: one-to-many, many-to-one, many-to-many, and derived configurations. In the one-to-many configuration, one physical sensor corresponds to many virtual sensors. In the many-to-one configuration, a virtual sensor communicates with several underlying physical sensors. Many-to-many configuration is a combination of the one-to-many and many-to-one configurations. In the derived configuration, the virtual sensor communicates with multiple sensor types but in the other three configurations, the virtual sensor communicates with the same type of physical sensors.

Also, authors of [5] use the virtual sensors. They believe that Sensor-Cloud infrastructure virtualizes a physical sensor as a virtual sensor on the cloud. They propose virtual sensor and virtual sensor group; each virtual sensor is created from one or more physical sensors and a virtual sensor group consists of one or more virtual sensors.
In [7], [11], [12] authors use a broker as an interface between sensors and users or applications. In proposed infrastructure of [11] and [12] Publish/Subscribe Broker (Pub/Sub Broker) is used. In the scenario that raised by the authors of [11], [12], users are connected to the cloud and subscribe data access requests. According to the framework that shown in Figure2, the requests are received via a component in the cloud. After creation of subscriptions, all of them are forwarded to the Pub/Sub Broker. Sensor data by way of gateway comes into the cloud. A processing component receives data from the gateway, identifies data, creates a published data event and sends the index of the data to the Pub/Sub Broker. When a subscription request reaches to the broker, it will start event matching. If broker finds any subscription that matches with published sensor data, it starts to retrieve data and then delivers the events to appropriate subscribers for use. Authors of [11] propose an event matching algorithm in the broker.

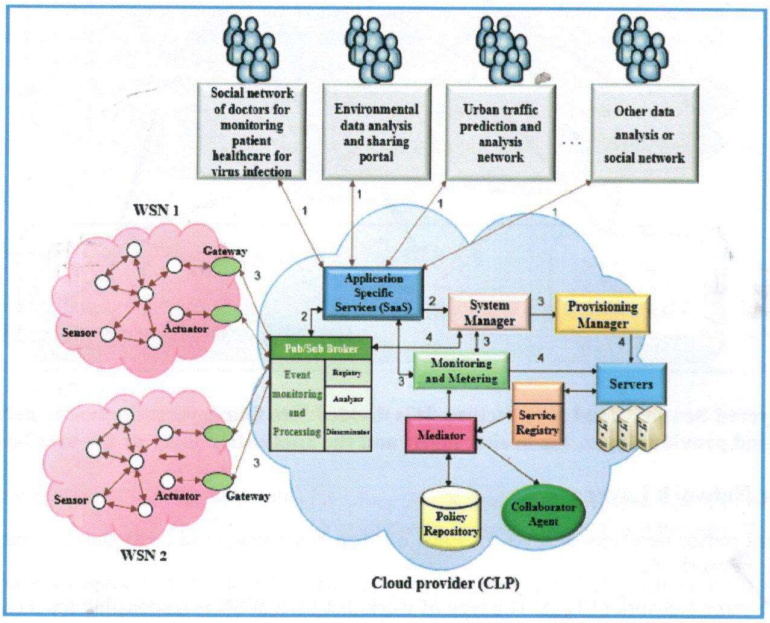

Figure2. A framework of Sensor-Cloud integration [12]

In [7] a broker is like an application in the cloud layer. Users and the sensor network owners (the sensor providers) register in Sensor Brokering Platform (SBP), then SBP send SLA template to both. Sensor provider fills template according to services that he wants to provision to the user. Users also fill template; indeed, they create request. SBP get the filling template from both sensor provider and user. The templates that have been received from sensor provider are stored. When SBP gets a request from the user, performs the lookup operation for finding service that meets SLA requirement of that user. If SBP finds no matching service, the request is rejected, and the negotiation process is repeated; but if the request is compliant to provided service, a new SLA will be signed. This new SLA summarizes the features that user has requested and also accepted.

3. Four-Layered Architecture of Sensor-Cloud

As previously mentioned, we propose a layered Sensor-Cloud architecture. As shown in Figure 3 Sensor-Cloud architecture consist of four layers:

1) Sensor network layer,

2) Cloud provider layer,
3) Broker layer and
4) Application layer or user layer.

Each layer in this architecture consists of multiple entities. For example, the sensor network layer consists of multiple WSNs, the cloud provider layer consist of multiple cloud providers and so on.

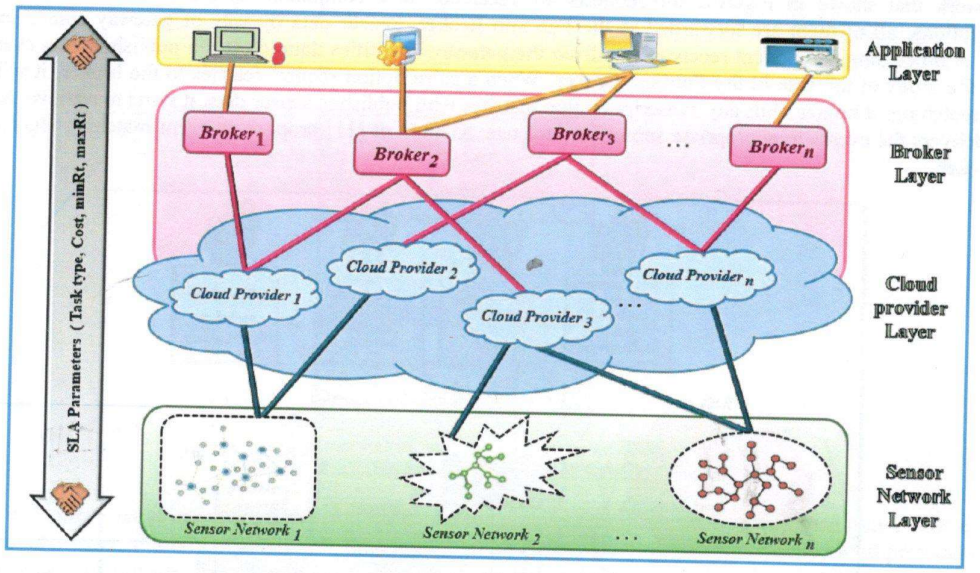

Figure 3. A layered Sensor-Cloud architecture. It is divided into four layers: the sensor network layer, the cloud provider layer, the broker layer, and the application layer or the user layer.

3.1. The Sensor Network Layer

There are different sensor networks in layer 1. Each of them has a unique id (uid) and do a particular task. SLA parameters of this network are:

1) Task Type of Sensor Network (TT_{SN}): The type of work that each WSN is responsible for doing it. According to Figure 3 we consider two task types for sensor networks.

$$\text{Task type of Sensor Network} = TT_{SN} = \{\text{Area monitoring, Health care monitoring}\} \quad (1)$$

2) Cost of Sensor Network (C_{SN}): The owner of a sensor network spend money on deploying sensors, programming and maintenance of sensors and so on. Thus a sensor provider must capture expense for providing services to others. In fact, this represents the cost that owner receives for service delivery. The cost of sensor network layer is defined in Eq. (2).

$$\text{Cost of Sensor Network} = C_{SN} \mid X \le C_{SN} \le Y,\ X = x\ (\$)\ ,\ Y = y(\$),\ X < Y \quad (2)$$

3) Response Time of Sensor Network (Rt_{SN}): For each of sensor networks, the response time has a lower and an upper bound. The lower bound is the minimum time that is spent from the received request until delivering the first response. Subsequently, the upper bound is the maximum acceptable time that is spent from the received request until delivering the first response. So the actual response time can be a number between these two values.

$$Rt_{SN} = [\min Rt_{SN}, \max Rt_{SN}] \quad (3)$$

To better express, the response time of the sensor network can be indicated by the following equations:

$$\text{Response time of Sensor Network} = Rt_{SN} \mid X \le Rt_{SN} \le Y,\ X = \text{minimum Response time of Sensor Network},\ Y = \text{maximum Response time of Sensor Network},\ X < Y \quad (4)$$

$$\text{minimum Response Time of Sensor Network} = \min Rt_{SN} \mid X \le \min Rt_{SN} \le Y,\ X = x\ (ms),\ Y = y\ (ms),\ X < Y \quad (5)$$

$$\text{maximum Response Time of Sensor Network} = \max Rt_{SN} \mid X \le \max Rt_{SN} \le Y,\ X = x\ (ms),\ Y = y\ (ms),\ X < Y \quad (6)$$

3.2. The Cloud Provider Layer

As depicted in Figure 3 there are multiple cloud providers in layer 2. Each of them has an uid. The owner of the sensor network can give their data to one or more cloud providers. Hence cloud providers must pay to the owner of the sensor network; this is the cost of the sensor network in layer 1. When a sensor network gives data to a cloud provider, in addition to the cost announces task type and response time to it.
Each of cloud providers has own SLA parameters including:

1) Cost of cloud provider (C_{CP}): This SLA parameter is the cost that cloud provider spend for storage and processing sensor network's data, creating and maintaining of the physical machines, virtual machines, virtual sensors and so on. We show this parameter in the Eq. (7).

$$\text{Cost of Cloud Provider} = C_{CP} \mid X \le C_{CP} \le Y,\ X = x\ (\$),\ Y = y\ (\$),\ X < Y \quad (7)$$

2) Response time of Cloud Provider (Rt_{CP}): It is time for processing function and so on in each of cloud providers. As shown in Eq. (8) this parameter like Rt_{SN} is characterized with upper and lower bound.

$$Rt_{CP} = [\min Rt_{CP}, \max Rt_{CP}] \quad (8)$$

And as a result:

$$\text{Response time of Cloud Provider} = Rt_{CP} \mid X \le Rt_{CP} \le Y,\ X = \text{minimum Response time of Cloud Provider},\ Y = \text{maximum Response time of Cloud Provider},\ X < Y \quad (9)$$

$$\text{minimum Response Time of Cloud Provider} = \min Rt_{CP} \mid X \le \min Rt_{CP} \le Y,\ X = x\ (ms),\ Y = y\ (ms),\ X < Y \quad (10)$$

$$\text{maximum Response Time of Cloud Provider} = \max Rt_{CP} \mid X \le \max Rt_{CP} \le Y,\ X = x\ (ms),\ Y = y\ (ms),\ X < Y \quad (11)$$

3.3. The Broker Layer

There are some brokers in layer 3. A broker also has an uid. As seen in Figure 3 each of brokers is connected to the cloud providers and applications. The broker stores SLA parameters for each of cloud providers that are connected to it. Another task that broker must do is receiving requests from the application that is connected to it. Then according to SLA parameters exist in the request of the application, the broker should deliver the best possible alternative to it. Each broker consists of a receiver component and a computational component. The receiver component receives the request of application from its top layer. The computational component with the use of MADM method gives the best possible alternative to the application. The alternative is actually one of cloud providers. This will be explained in detail later in this paper.

3.4. The Application Layer

Applications or users are in layer 4. Each user for satisfying his demand (task type), can pay a maximum amount of money. The task type and the cost of application are shown in Eq. (12) moreover, Eq. (13).

$$\text{Task type of Application} = TT_{App} = \{\text{Area monitoring, Health care monitoring}\} \quad (12)$$

$$\text{Cost of Application} = C_{CP} \mid X \leq C_{CP} \leq Y,\ X = x\ (\$),\ Y = y\ (\$),\ X < Y \quad (13)$$

Each user can wait a time, from send request to receive a response. This time is shown in the Eq. (14).

$$Rt_{App} = [minRt_{App}, maxRt_{App}] \quad (14)$$

As a result, we have:

$$\text{Response time of Application} = Rt_{App} \mid X \leq Rt_{App} \leq Y,\ X = \text{minimum Response time of Application},\ Y = \text{maximum Response time of Application},\ X < Y \quad (15)$$

$$\text{minimum Response Time of Application} = minRt_{App} \mid X \leq minRt_{App} \leq Y,\ X = x\ (\text{ms}),\ Y = y\ (\text{ms}),\ X < Y \quad (16)$$

$$\text{maximum Response Time of Application} = maxRt_{App} \mid X \leq maxRt_{App} \leq Y,\ X = x\ (\text{ms}),\ Y = y\ (\text{ms}),\ X < Y \quad (17)$$

It is possible that two or more brokers are connected to the same cloud provider. Each of brokers delivers one alternative to the applications that are connected to it. This alternative contains the uid of the cloud provider, the uid of the broker, and the total cost and total response time. The user receives alternatives from brokers that are connected to them and then, he selects the best one among received alternatives.

3.5. Scenario

In order that demand of each application is satisfied; first, the sensors should sense data from the environment. The sensors are not directly linked to the applications. So WSNs send data to their upper layer. As a result, data and also SLA parameters of each WSNs is available to one or more cloud provider. In Figure4 we show one of the possible configurations, this figure represents concepts that we proposed.

The cloud providers process aforementioned data and store if necessary. So performing the task of application requires spending cost and time in both layers 1 and 2. In other words, the cost consists of two parts: the cost that cloud provider receives from users, for providing different services to them and, the cost that it pays to the owner of the WSN. This parameter is named total cost. In fact, a user must pay both costs, WSN owner's cost and cloud provider's cost. The concepts that have been explained shown in the
Figure5. As is evident in the figure, the cloud provider with the uid 3 is connected to sensor network with the uid 2. The cloud provider 3 collect its cost (212) with sensor network's cost (323), and then sends the sum of costs (535) to a broker that is connected to it (broker 0).
Subsequently, there is a total response time for each connection between the WSN and the cloud provider. It is the sum of WSN's response time and cloud provider's response time that WSN is connected to it. Total response time also has two upper and lower bounds. The lower bound is Total of minimum Response time ($minRt_{Tot}$) that is a summation of $minRt_{SN}$ and $minRt_{CP}$. The upper bound is Total of maximum Response time ($maxRt_{Tot}$). $maxRt_{Tot}$ is easily calculated by adding the number of $maxRt_{SN}$ and $maxRt_{CP}$. Each of cloud providers uses the following equations for each WSN that is connected to it.

$$\text{Task Type for Sensor Cloud relation} = TT_{ij} \mid TT_{ij} = \text{Task Type of Sensor Network i that is connected to Cloud Provider j} \quad (18)$$

$$\text{Total Cost for Sensor Cloud relation} = C_{Tot_{ij}} \mid C_{Tot_{ij}} = C_{SN_i} + C_{CP_j} \quad (19)$$

$$\text{Total Response Time for Sensor Cloud relation} = Rt_{Tot_{ij}} \mid Rt_{Tot_{ij}} = [minRt_{Tot_{ij}}, maxRt_{Tot_{ij}}] \quad (20)$$

Total of minimum Response Time for Sensor Cloud relation = $\min Rt_{Tot_{ij}} \mid \min Rt_{Tot_{ij}} = \min Rt_{SN_i} + \min Rt_{CP_j}$ (21)

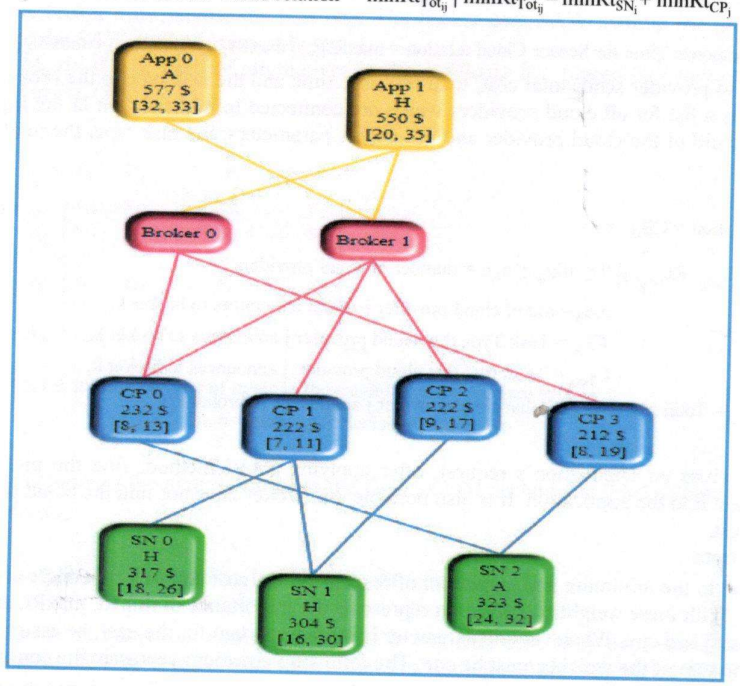

Figure4. Configuration and connection of Sensor Network (SN), Cloud Provider (CP), Broker and Application (App) and related SLA parameters

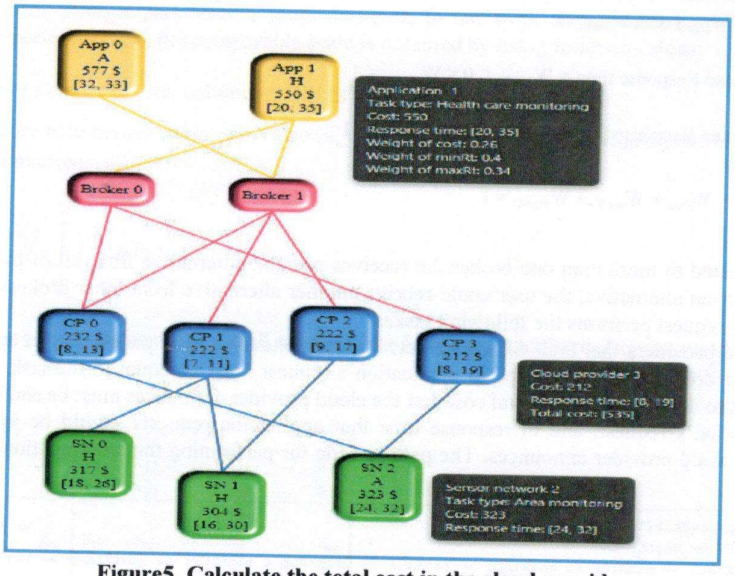

Figure5. Calculate the total cost in the cloud providers

Total of maximum Response Time for Sensor Cloud relation = $maxRt_{Tot_{ij}} \mid maxRt_{Tot_{ij}} = maxRt_{SN_i} + maxRt_{CP_j}$ (22)

Eventually, the cloud provider sends total cost, total response time and the task type to the brokers that is connected to it. The broker has a list for all cloud providers which are connected to it. As shown in the Eq. (23) each item of the list includes the uid of the cloud provider and three SLA parameters: the task type, the total response time and the total cost.

Cloud broker relation = CB_{jk} =

$\{ (uid_{jk}, TT_{jk}, C_{Tot_{jk}}, Rt_{Tot_{jk}}) \mid 0 \leq uid_{jk} \leq n$, n = number of cloud providers,

uid_{jk}= uid of cloud provider j, that it announces to broker k,

TT_{jk} = Task Type that cloud provider j announces to broker k,

$C_{Tot_{jk}}$= Total cost that cloud provider j announces to broker k,

$Rt_{Tot_{jk}}$= Total response time that cloud provider j announces to broker k $\}$ (23)

Once a broker receives an application's request, after applying SAW method, find the most appropriate cloud provider and suggest it to the application. It is also possible that broker does not find the cloud provider according to application's request.

As is evident from

Figure5, in addition to the minimum and maximum of response time, cost and task type there are weights in request of the application. With these weights, application represents the importance of minRt, maxRt, and cost. Weight is a number between zero and one. Whatever one parameter is more important for the user, he assigns a larger number to that parameter. The sum of the weights must be one. The following equations represent the concepts we explained.

$W = \{W_{cost}, W_{minRt}, W_{maxRt}\}$ (24)

Weight of cost = $W_{cost} \mid 0 \leq W_{cost} \leq 1$ (25)

Weight of minimum Response time = $W_{minRt} \mid 0 \leq W_{minRt} \leq 1$ (26)

Weight of maximum Response time = $W_{maxRt} \mid 0 \leq W_{maxRt} \leq 1$ (27)

$$\sum W = W_{Cost} + W_{minRt} + W_{maxRt} = 1 \quad (28)$$

If the user is connected to more than one broker, he receives possible alternative from all of them. If one or more brokers did not offer an alternative, the user could receive another alternative from other brokers. Each broker after receiving the user's request performs the following tasks:

1) First, finds cloud providers that their SLA parameters corresponded to SLA parameters in application's request (candidate list). For doing this, a) task type in application's request must be equal to the task type that the cloud provider announces to the broker, b) the total cost that the cloud provider announces must be smaller than or equal to the cost in application's request, and c) response time that application requests should be in the range of total response time that cloud provider announces. The pseudo code for performing the aforementioned task is shown in Figure6.

```
if (taskTypeApp.equals(taskTypeCP) &&
        costAPP >= costCP &&
        ( minRtApp >= minRtCP && maxRtApp >= maxRtCP || minRtApp < minRtCP) && (maxRtApp >= maxRtCP ) ) {
      add this cloud provider to candidate list;
}
```

Figure6. Pseudo code that the broker using for finding the candidate cloud providers

2) Now the broker has a list of cloud providers that can meet application's request. If the number of items in the list is more than one, the broker must choose the best one. The broker uses SAW method for finding the best cloud provider. The steps of the SAW method are as follows:
Construct Decision Matrix (DM): Each of alternatives in the candidate list, placed in a row of DM. The decision matrix is represented in Eq. (30):

$$DM = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_i \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccccc} P_1 & P_2 & \cdots & P_j & \cdots & P_n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} & \cdots & x_{mn} \end{bmatrix}_{m\times n} \end{array}$$

$1 \le i \le m$, m = number of items in the candidate list (alternatives),
$1 \le j \le n$, n = number of decision parameters (30)

In our work, the first column of the matrix indicates the C_{Tot}, and the second and third columns represent $minRt_{Tot}$ and $maxRt_{Tot}$ respectively.

$$n = 3,\ P_1 = C_{Tot}\ (\$),\ P_2 = minRt_{Tot}\ (ms),\ P_3 = maxRt_{Tot}\ (ms) \quad (31)$$

2-1) Construct the normalized decision matrix: As is clear in Eq. (31) each SLA parameters (each column of the matrix) has their scale, which may conflict with each other. For finding the best alternative in the candidate list, we should transform the dimensions of various parameters into nondimensional parameters, which allows comparison across the parameters [13]. For this purpose, SAW uses linear scale transformation method. In this method, we must recognize that whether the type of parameter is benefit or cost. For the cost parameters, a smaller parameter, and for the benefit parameters, a larger parameter is more desirable. In our work all parameters are the cost criteria; the smaller, the higher preference. So, the comparable scale is obtained by using following steps:

a) Find the minimum number in each column of DM as x_n^- and,

b) A simple procedure is to divide minimum value (x_n^-) by the outcome of an individual criterion. Eq. (32) represent linear scale transformation:

$$r_{ij} = {x_j^-}/{x_{ij}} \quad \Big| \quad x_j^- = \min_i x_{ij} \quad (32)$$

By applying a linear scale transformation on all entries in the DM, the normalized decision matrix has been achieved as follows:

$$DM' = \begin{array}{c} \begin{array}{cccccc} P_1 & P_2 & \cdots & P_j & \cdots & P_{mn} \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} & \cdots & x_{mn} \end{bmatrix}_{m\times n} \end{array} \quad (33)$$

2-2) The weight vector that application announced should be effective in finding the best alternative. So, according to Eq. (34) each entry in DM' is multiplied by the weight vector:

$$DM' \times \text{Weight Vector} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}_{m\times n} \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}_{n\times 1} =$$

$$\begin{bmatrix} r_{11} * w_1 + r_{12} * w_2 + \ldots + r_{1n} * w_n \\ r_{21} * w_1 + r_{22} * w_2 + \ldots + r_{2n} * w_n \\ \vdots \\ r_{m1} * w_1 + r_{m2} * w_2 + \ldots + r_{mn} * w_n \end{bmatrix}_{m\times 1} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_m \end{bmatrix}_{m\times 1} \quad (34)$$

In other words, for each item in the candidate list we have:

$$S_i = \sum_{j=1}^{n} W_j \, r_{ij} \;\Big|\; n = \text{number of Decision Parameter}, \; 1 \le i \le m,$$
$$m = \text{number of Cloud Provider that meets SLA requirement of Application} \quad (35)$$

The S_i represent scores of each alternative, in fact after applying SAW method each item in the candidate list have a score, according to Eq. (36) the row which has the biggest score among other, have been selected as the best alternative.

$$S^* = \max_i S_i \;\Big|\; 1 \le i \le m, \; m = \text{number of member in candidate list} \quad (36)$$

Finally, the item with the highest score will be sent to the application. This item includes the uid of a cloud provider, the uid of a broker, the total cost and the total response time. The application receives one or more than one item. When an application receives more than one item, it must communicate with multiple brokers. If an application receives one item, selects it, but if it receives more than one item it again applies SAW method over receiving items to find the best item. Eventually, the application selects an item (the cloud provider) with the highest score. The approach that we used in the scenario of Section 3 is a local approach. In this scenario, there are multiple brokers in different configurations. Each of them connects to one or more than one cloud provider. The application that connects to more brokers receives better offers. We provide more details about this topic in the next section.

4. Analysis

In this section, we present our simulation results as well as analysis of our work. We have used some configurations in our work. We implemented the SAW method with Java. In the local approach, all of the brokers do the steps as follows: The broker k for each application that is connected to it, call SAW method if the candidate list of the broker k is not empty (as explained section 3.5). This method returns the best item to the caller. Then the brokers send the best item to the applications. Also if an application receives more than one item from multiple brokers, call SAW method.

For all users, the ideal occurs when there is a broker that can connect to all cloud providers in layer 2. For this reason, we assume a global approach. In this approach, there is only one broker in layer 3. This broker can communicate with all cloud providers and all applications. The attitude above requires that broker has high communication and computational power. In the global approach, if the candidate list of the broker is not empty, certainly the best alternative is found for all users. Because of an item which has been found by this broker is certainly the best one, so running the SAW method one more time no longer is necessary. So in our implementation, in the global algorithm, none of the application call the SAW method. In the following, we ran the algorithm on different configurations and compared the global and local approaches.

To compare the local and the global approaches we did as follow:

- ✓ Executed both the local and global approaches several times on one configuration,
- ✓ Considered the answers that each application received in the Local approach: A_L,
- ✓ Considered the answers that each application received in the Global approach: A_G,
- ✓ In each execution, examined whether A_L equals to A_G ($A_L = A_G$) or not,
- ✓ And finally counted the Number of times that $A_L = A_G$ (N_{EQ}).

We executed the local and global approaches 300 times on Figure4 configuration. Then without changing the other components and connections, increased the number of sensors, and again executed both approaches 300 times for each of configurations. As shown in Figure4, the application 0 is connected to the broker 1, and the application 1 is connected to the broker 0 and broker 1. The Figure7 represents N_{EQ} in 13 configurations. As it is clear, in all cases, N_{EQ} of application 1 is more than N_{EQ} of application 0. In other words, whereas application 1 is connected to the both existing brokers can receive the better answer than application 0.

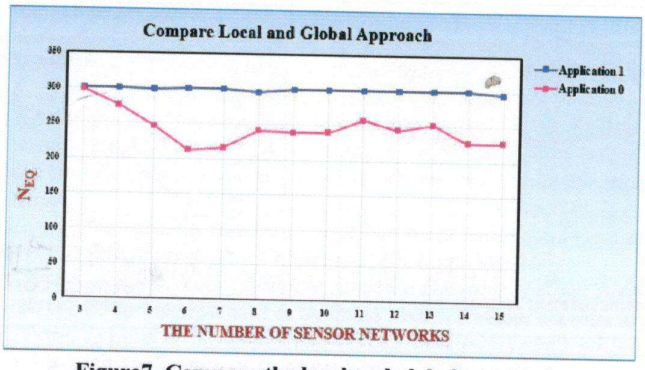

Figure7. Compare the local and global approach

In the left configuration of Fig 8 we show one of possible configurations. By keeping constant all components included in the configuration, we only changed the number of connections in the applications. First, all applications connected to only one broker and the local and global approaches executed 1000 times. Then in all applications, the number of connections increased to two, three, four and five connections and both approaches executed again.

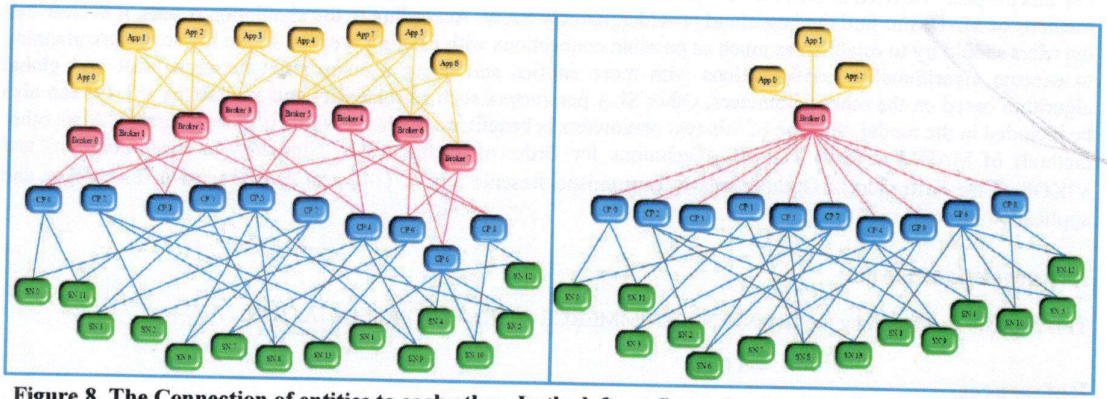

Figure 8. The Connection of entities to each other. In the left configuration, each application is connected to three brokers (local approach). In the right configuration, there is one broker in layer 3 (global approach)

The simulation results is shown in Fig 8. the number of connections for each application is three The simulation results show that whatever the applications are connected to more brokers can receive better offers.

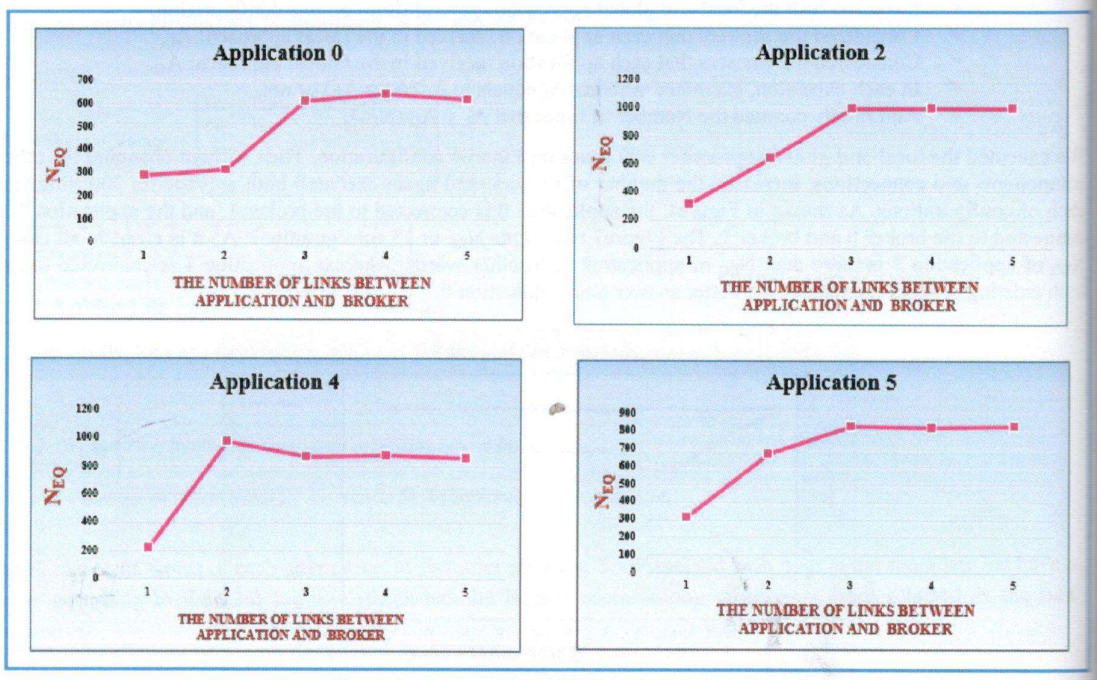

Figure9. Compare the local and global approach

5. Conclusion and Future Works

In this paper, we proposed a four-layered Sensor-Cloud architecture that contains the sensor network layer, the cloud provider layer, the broker layer, and the application layer. Our goal was to handle SLA in four-layered architecture. For this purpose, we tried to find the best possible level of SLA for each user. Therefore, we implemented one of the methods of MADM to find the best cloud provider, namely SAW. According to the simulation results, it is clear that the users should try to establish as much as possible connections with existing brokers. In the future, we are planning to execute algorithms on configurations with more entities and doing a comparison between local and global algorithm based on the other parameters. Other SLA parameters such as reliability and availability [5], [9] can also be included in the model, the type of this two parameters is benefit, so the larger value is more desirable. Also, other methods of MADM such as TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) and VIKOR (Vlse Kriterijumsk Optimizacija Kompromisno Resenje) [13], [14], etc. can be used in the broker and application layer.

Acknowledgements

This paper is sponsored by the IRANIAN E-COMMERCE SCIENTIFIC ASSOCIATION.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
[2] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.
[3] S. Madria, V. Kumar, and R. Dalvi, "Sensor Cloud: A Cloud of Virtual Sensors," *IEEE Software*, vol. 31, no. 2, pp. 70–77, 2014.

[4] Yinbiao, S. "Internet of things: wireless sensor networks." White Paper, International Electrotechnical Commission, Available: http://www.iec.ch/whitepaper/internetofthings, 2014.
[5] M. Yuriyama and T. Kushida, "Sensor-Cloud infrastructure physical Sensor Management with Virtualized Sensors on Cloud Computing," in *Network-Based Information Systems (NBiS), 13th International Conference*, 2010, pp. 1–8.
[6] S. K. Dash, J. P. Sahoo, S. Mohapatra, and S. P. Pati, "Sensor-Cloud: Assimilation of Wireless Sensor Network and the Cloud," *Advances in Computer Science and Information Technology. Networks and Communications*, vol. 84, no. PART 1, pp. 455–464, 2012.
[7] V. Casola, A. De Benedictis, M. Rak, G. Aversano, and U. Villano, "An SLA-based brokering platform to provide sensor networks as-a-service," *International Journal of Business Process Integration and Management*, vol. 7, no. 2, pp. 114–127, 2014.
[8] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. A. Hossain, A. Alelaiwi, and M. S. A. Hossain, "A survey on Sensor-Cloud: Architecture, Applications, and Approaches," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 917923, 18 pages, 2013.
[9] L. Z. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.
[10] D. Ardagna, C. Ghezzi, and R. Mirandola, "Model Driven QoS Analyses of Composed Web Services," in *European Conference on a Service-Based Internet, 1st European Conference*, 2008, pp. 299–311.
[11] M. M. Hassan, B. Song, and E.-N. Huh, "A Framework of Sensor-Cloud Integration Opportunities and Challenges," in *Conference on Ubiquitous Information Management and Communication, the 3rd International Conference*, 2009, pp. 618–626.
[12] K. Ahmed and M. Gregory, "Integrating Wireless Sensor Networks with Cloud Computing," in *Mobile Ad-hoc and Sensor Networks (MSN), Seventh International Conference*, 2011, pp. 364–366.
[13] M. Economics, C.-L. Hwang, and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications a State-of-the-Art Survey*. New York: Springer Science & Business Media, 2012.
[14] Y. Shi, S. Wang, Y. Peng, J. Li, and Y. Zeng, *Cutting-Edge Research Topics on Multiple Criteria Decision Making*. New York, NY, USA: Springer Science & Business Media, 2009.