International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 2, pages 218–239, July 2018

A Multi-agent Based Flexible IoT Edge Computing Architecture Harmonizing Its Control with Cloud Computing

Tadashi Ogino
School of Information Science, Meisei-University
Hino, Japan

Shinji Kitagami
Faculty of Environmental and Information Sciences, Fukui University of Technology
Fukui, Japan

Takuo Suganuma
Cyberscience Center, Tohoku University
Sendai, Japan

Norio Shiratori
Research and Development Initiative, Chuo University
Tokyo, Japan

Received: January 17, 2018
Revised: April 17, 2018
Accepted: June 6, 2018
Communicated by Yoshiaki Kakuda

Abstract

Recently, in large-scale Internet of Things (IoT) systems based on cloud computing, problems such as increase in network load, delay in response, and invasion of privacy have become concerning. To solve these problems, edge computing has been introduced to the IoT systems. However, if the cloud function is excessively migrated to the edge, the collected data cannot be shared between IoT systems, thus reducing the system's usefulness. In this paper, we propose a multi-agent based flexible IoT edge computing architecture to balance global optimization by a cloud and local optimization by edges and to optimize the role of the cloud server and the edge servers dynamically. Further, as an application example, we introduce an energy management system based on the proposed edge computing system architecture to demonstrate the effectiveness of our proposal.

Keywords: IoT, edge computing, cloud computing, flexible, multi-agent

1 Introduction

Recently Internet of Things (IoT) systems, in which many sensors or devices are connected directly to the Internet to provide various services without human intervention, have been attracting attention [1–3]. IoT applications are adopted in the industrial, household, as well as social sectors. In the industrial sector, IoT application provides an increasing sophistication of remote maintenance and supply chain management. In the home sector, it offers well-developed health care and energy management. In the social sector, IoT systems are effective for disaster prevention, such as monitoring and preventing floods. These conventional IoT systems are based on a cloud-centric architecture. Therefore, problems such as increase in network load, delayed feedback response, and invasion of privacy are identified in a large-scale IoT system [4].

To solve these problems, the concept of edge computing (EC) has been introduced to the IoT architecture [4, 5]. In EC, the data collection, filtering, and feedback control functions are implemented on the edge servers in the base-stations of the mobile communication carrier or the IoT gateways close to the sensors and actuators. EC is effective in solving communication traffic shortage and delayed feedback control issues. However, if the cloud function is excessively migrated to the edge, the collected data cannot be shared between IoT systems and this decreases the system's usefulness [6]. Moreover, while EC is effective for local optimization in an edge domain, it does not aid in global optimization of multiple domains.

Studies have been conducted to address these problems. A previous research [7] proposed a method to cooperate analysis of the data distributed to clouds and edges in the electric power system. Another research [8, 9] proposed an environment adaptive IoT architecture to optimize the roles of clouds and edges.

In this paper, we extend these previous studies and propose a multi-agent based flexible IoT-EC architecture to solve the problems of the conventional EC. The proposed IoT architecture balances global optimization by a cloud and local optimization by edges to optimize the roles of the cloud server and the edge servers dynamically using multi-agent technology. In Section 2, we present the background of the research. In Section 3, we propose the concept of flexible IoT-EC and its architecture. In Section 4, as an application example, we introduce an energy management system and demonstrate the effectiveness of our proposed architecture. In the field of energy management systems, responding to imminent power supply shortage and realizing efficient power consumption has become a problem. Thus, a demand response (DR) system has been developed that introduces a smart meter with a bidirectional communication function in the wattmeter and efficiently operates the entire power grid while exchanging information on power supply and power consumption. A DR system is a large-scale IoT system with smart meters as IoT devices. In the current DR system, power consumption is reduced after informing the customer of the reduction in power supply. However, the customer's situation is not considered in this system. To operate the entire power grid system efficiently, it is necessary to properly balance information and processes on the supply side (cloud) and the customer side (edge). In our example, we demonstrate how our proposed IoT architecture can be used to improve the DR system.

2 Background of This Research

2.1 Conventional Cloud-centric IoT Architecture

Various IoT architectures are proposed by standards bodies and researchers [1, 3]. A couple of architectures are based on a three-layer IoT architecture, as shown in Figure.1(a). As functional decomposition depends on the architecture, this three-layer architecture is regarded as vertical integration IoT architecture. Another type of architecture is the five-layer IoT architecture that extracts the common function from the three-layer IoT architecture and adds a business layer to it. Figure.1(b) is one example of a five-layer IoT architecture proposed by Al-Fuqaha called API-based five-layer IoT architecture.

The service management layer of the five-layer IoT architecture supports common IoT functions, such as data collection, data analysis, device management, and service discovery. These common functions are considered horizontal integrated IoT platform on the cloud. As a result, certain advantages such as reduction of development cost, protocol independency, and easy reuse of collected data can be achieved.

In the five-layered IoT architecture, all the data is collected and analyzed on the cloud. Further, all the actuators in the object layer are controlled by the results on the cloud. This behavior causes

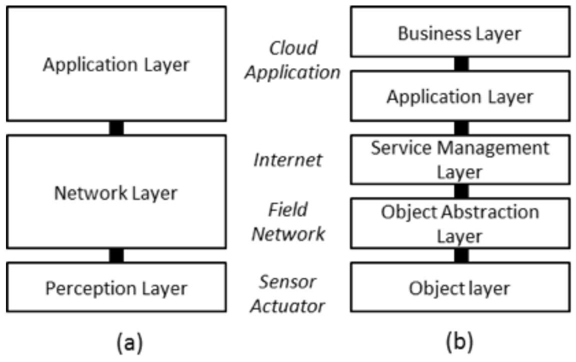


Figure 1: IoT Architecture: a) Three-layer IoT Architecture; b) Five-layer IoT Architecture

some demerits [4].

1) Under a large IoT system containing several sensors, collecting a large amount of data leads to communication traffic shortage and deteriorates the service.

2) The communication convergence in the Internet and the cloud causes control delay. The system delay also depends on the frequency of the data collection.

3) Collecting all the data on the cloud causes more serious security issues.

2.2 IoT Edge Computing

To solve the problems explained in the previous section, the concept of EC is introduced to the IoT architecture [5, 6, 10–21]. EC is the method of performing the processing at the place near the data origin or control targets. A well-known example is a content delivery network (CDN). A CDN deploys the Internet contents near the clients to improve Web performance.

There are various types of EC architectures including three-layer, four-layer, and seven-layer. The most common EC architecture is the three-layer architecture that adds an edge layer between the device and cloud layers. Figure.2 shows the IoT architecture with a three-layer EC system. In this architecture, the data collection, filtering function, and feedback control functions are implemented on the edge servers in the career base-stations or IoT gateways closer to the IoT devices.

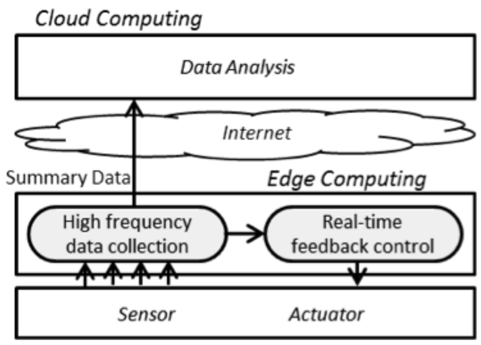


Figure 2: IoT-EC

T. G. Rodrigues *et al.* proposed a method to offload tasks from cloud to cloudlet servers in the edge [11]. They presented a mathematical model of their method.

J. Ren *et al.* proposed a transparent computing based IoT architecture [12]. Under this architecture, the required services are loaded on the edge servers from the cloud. As a result, on-demand apps/services can be dynamically provisioned for lightweight IoT devices.

J. Ren also proposed mobile crowdsourcing that can enable mobile users to acquire the outsourced task. By leveraging the sensing capabilities of mobile devices and integrating human intelligence with machine-computation, mobile crowdsourcing has the potential to revolutionize the approach to data collecting and processing [14].

M. Marjanović *et al.* proposed an EC architecture adequate for massive-scale mobile crowd sensing (MCS) services [22]. MCS is a human-driven IoT service that empowers citizens to observe the phenomena of individual, community, or even societal values by sharing sensor data about their environment while on the move [23]. By moving computation to the network edge, the associated traffic in the mobile core is reduced. Sahni *et al.* proposed a new computing paradigm, named Edge Mesh, which distributes the decision-making tasks among edge devices within the network instead of sending all data to a centralized server [24]. RedEdge is a big data processing solution that enables processing of big data streams near the data source in mobile edge cloud computing environments [25]. In particular, they focus on a new dimension that IoT adds to big data and analytics: a massively distributed number of sources at the edge.

Fog computing is a similar concept to transparent computing. Proposed by F. Bonomi, it is a hierarchical distributed architecture that extends from the edge of the network to the core [26]. T. H. Luan *et al.* presented a three-tier Mobile-Fog-Cloud architecture that deploys highly virtualized computing and communication facilities with easy access to mobile users [27]. B. Tang *et al.* introduced a four-layer hierarchical fog computing architecture for big data analysis in smart cities [28]. It parallelizes data processing at the edge of the network, which satisfies the requirements of location awareness and low latency for smart city services.

2.3 Problems of IoT Edge Computing

IoT-EC is effective in solving network traffic shortage and delay in feedback control. However, there are a couple of problems in IoT-EC as described below [6, 9].

Problem-1) Provisioning of the IoT functions depends on the resources and network environment of the edge servers. Under the large-scale IoT system, it is difficult to deploy the same resources for all edge servers. That is, we need a method to optimize all IoT systems by changing the roles of the cloud and the edge part dynamically according to the resources and the network environment of the edge servers.

Problem-2) If all the IoT functions are placed at the edge servers, all the IoT systems become the localized vertical integrated system. This prevents global optimization based on the collected data. On the contrary, prioritizing global optimization in cloud hinders local optimization, such as real-time control in edge. That is, when we introduce EC to the IoT system, we need a mechanism to balance global optimization by a cloud and local optimization by edges.

Problem-3) In IoT-EC, local optimization at one edge domain may interfere with local optimization of other edge domains. In that case, a mechanism is needed to coordinate the edges and to balance local optimization of edges without going through the cloud.

3 Flexible Multi-agent Based IoT Edge Computing

There have been consistent studies [6–9] that aim to solve the problems of IoT-EC described in the previous section. To solve Problem-1) of IoT-EC, an environment adaptive IoT architecture was proposed in the studies [6, 7]. Figure.3 shows the concept of the environment adaptive IoT architecture. In the figure, “Environment Adaptability” autonomously assigns the allocation of processing to edges and clouds as appropriate points on a plane by two axes according to the quality of work, quantity, and variation. On the other hand, “User-oriented Property” is reflected in the appropriate service for each user by autonomous control based on the user’s behavior collected by IoT and on detailed information such as expression and gesture.

In this paper, based on the concept presented in [6, 7], we propose a flexible IoT-EC architecture to solve the above Problem-2 and to determine a mechanism to balance the global optimization by cloud and the local optimization by edges. We also formulate the optimization and demonstrate the

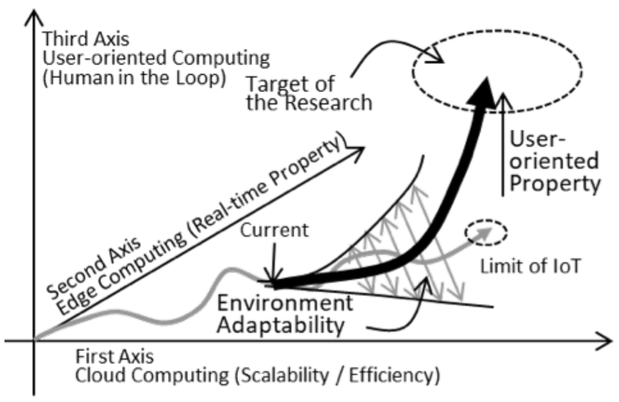


Figure 3: Concept of Environment Adaptive IoT Architecture

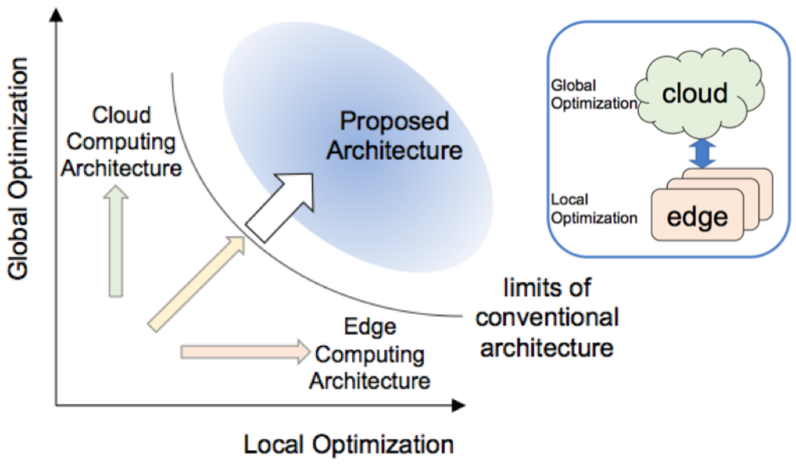


Figure 4: Balancing Cloud Performance and Edge Performance

intelligent protocol to satisfy the optimization. Thereafter, an example application for the energy management is described.

3.1 Basic Concept

Here, we will explain how total balancing mechanisms work in our proposed architecture.

The balancing optimization functions are divided into both the cloud side and the edge side. Each optimization subtask can only optimize its side because it does not have enough information to tend to the other side. The cloud side subtask, that is, global optimization subtask, can improve its performance of the cloud, but that does not improve the edge performance, as shown in Figure.4. For the edge subtask (local optimization subtask), the situation is reversed and it cannot improve the edge side performance. Especially, for actual applications described in the following section, a relationship between the cloud and the edges is often a trade-off relationship. By simply improving the performance of one side, the performance of the other side may be decreased.

In our architecture, we propose the balancing optimization mechanism with the collaboration of both the global and the local subtask. The goal of this mechanism is to achieve total optimization of the system, as shown in Figure.4.

The architecture, formulation, and protocol of our proposed mechanism are explained below.

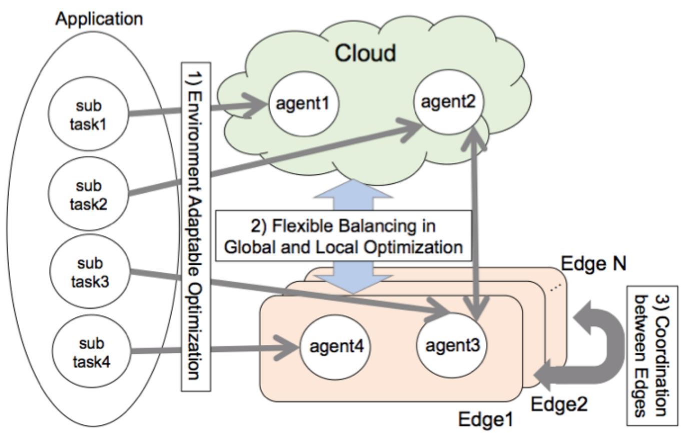

Application
sub task1
sub task2
sub task3
sub task4
1) Environment Adaptable Optimization
Cloud
agent1
agent2
2) Flexible Balancing in Global and Local Optimization
Edge N
agent4
agent3
Edge1
Edge2
3) Coordination between Edges

Figure 5: Multi-agent based Architecture of Flexible IoT-EC

3.2 Architecture

Figure.5 shows the overall proposed architecture. An application is divided into multiple subtasks that are assigned to a cloud or edges according to their characteristics as agents. For example, a global optimization subtask is placed in the cloud as it needs to access all the summary information gathered at the cloud. A real-time actuator control subtask, however, requires low latency communication and thus it is assigned to the edge servers connected to the actuators.

Subtasks typically use autonomous distributed multi-agency technology. When necessary, agents can move from the cloud to the edges, from one edge to another, etc. This is how Problem-1) is solved. The complete process is described in detail in [7].

When an application is divided into subtasks and distributed to the cloud and the edges, it is necessary to have a mechanism that allows the entire system to work properly. If all information that determines the behavior of the entire system is gathered in the cloud, then the optimization and control functions can also be executed only in the cloud. In many cases, such information is dispersed throughout the system and it is difficult for a single agent in the cloud to control the entire system. When such agents exist both in the cloud and the edges, both agents should collaborate so that the system balances properly for both the cloud and edge. In this paper, we will explore this optimization mechanism in 3.4. In some cases, one edge agent needs to communicate with a different edge agent in its vicinity. Alternatively, if necessary, the cloud may communicate with other clouds to collaborate a task. In our system, these functions are realized as the cooperation between edges or clouds. With the mechanisms described above, our proposed system can overcome three challenges encountered by IoT-EC.

3.3 Formulation

To balance the optimization of the total application, the cloud and edge subtasks should communicate and discuss the details of the optimization process.

When an application is divided into subtasks and deployed to the cloud and the edges, each subtask has limited access to the system information that is necessary to optimize the total system. As a result, a subtask in the cloud can optimize only the cloud system and subtasks in the edges can optimize only the edge systems.

Let us explain this situation with formalization.

1. Basic case

$$\begin{array}{ll} \text{variables:} & \boldsymbol{v} = [v_1, v_2, \cdots v_N] \\ \text{cost function:} & cost(\boldsymbol{v}) \\ \text{optimization:} & \underset{v_1, v_2, \cdots v_N}{\text{MIN}} (cost(\boldsymbol{v})) \text{ under } constraints(\boldsymbol{v}) \end{array} \tag{1}$$

All the parameters that affect the system behavior are described as variables $v_i (i = 1 \text{ to N})$. We merge various indicators that evaluate the system behavior into one evaluation function. Here, we aim to minimize the evaluation function. We call this evaluation function as a cost function. Though not all variables can be freely changed, there are some constraints (such as $0 < v_0 < v_1 + v_2$). Therefore, the system optimization is paraphrased as a problem of minimizing the cost function under certain constraints.

Variables $\boldsymbol{v}$ vary with time. The optimization subtask executes the calculation and selects the suitable values. Although whether the optimum value to minimize the cost function can be obtained mathematically is not discussed in this paper, we assume that some values of practical range can be obtained using a suitable method.

2. IoT-EC case

When an application is distributed to the cloud and edges, the variables, cost functions, and constraints vary depending on the node.

variables:

$$\begin{array}{ll} \boldsymbol{v_c} = [v_{c1}, v_{c2}, \cdots v_{cK}] & \text{; from cloud} \\ \boldsymbol{v_e} = [v_{e1}, v_{e2}, \cdots v_{eL}] & \text{; from edges} \\ \boldsymbol{v_s} = [v_{s1}, v_{s2}, \cdots v_{sM}] & \text{; from both} \end{array} \tag{2}$$

cost function:

$$\begin{array}{ll} cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}) & \text{; for cloud} \\ cost_e(\boldsymbol{v_s}, \boldsymbol{v_e}) & \text{; for edge} \\ cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}, \boldsymbol{v_e}) = & \\ cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}) + k * \sum_{\text{All edge}} cost_e(\boldsymbol{v_s}, \boldsymbol{v_e}) & \text{; for both cloud and edges} \end{array} \tag{3}$$

global optimization (cloud):

$$\underset{\boldsymbol{v_s}, \boldsymbol{v_c}}{\text{MIN}}(cost_c(\boldsymbol{v_s}, \boldsymbol{v_c})) \text{ under } constraints_c(\boldsymbol{v_s}, \boldsymbol{v_c}) \tag{4}$$

local optimization (edge):

$$\underset{\boldsymbol{v_s}, \boldsymbol{v_e}}{\text{MIN}}(cost_e(\boldsymbol{v_s}, \boldsymbol{v_e})) \text{ under } constraints_e(\boldsymbol{v_s}, \boldsymbol{v_e}) \tag{5}$$

total optimization:

$$\underset{\boldsymbol{v_s}, \boldsymbol{v_c}, \boldsymbol{v_e}}{\text{MIN}} (cost_t(\boldsymbol{v_s}, \boldsymbol{v_c}, \boldsymbol{v_e})) \text{ under } constraints_t(\boldsymbol{v_s}, \boldsymbol{v_c}, \boldsymbol{v_e}) \tag{6}$$

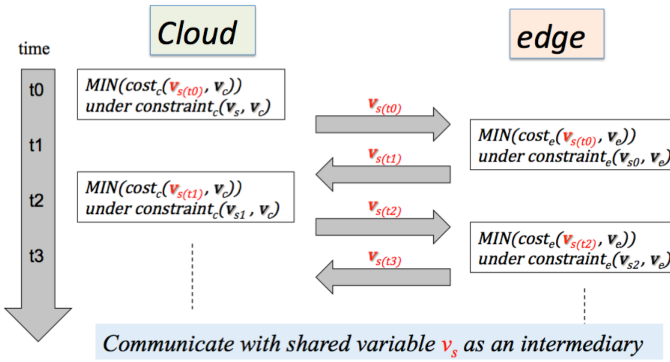
The variables $\boldsymbol{v_i}$ are classified according to accessible nodes: from the cloud, from the edges, and from both. $\boldsymbol{v_s}$ is a shared variable and is used to affect both the cloud and the edge system. $\boldsymbol{v_c}$ is a variable used to affect only the cloud behavior. $\boldsymbol{v_e}$ is a variable used to affect only the edge behavior.

Particularly, $cost_e()$ are different for each edge, but we use the same designation $cost_e()$ to make the expression easier to read. In the case of an IoT system in which data loss occurs frequently, the cost function must include those data loss situations.

$cost_c()$ and $cost_e()$ are calculated only in the cloud or the edges, respectively. As a result, the total optimization cannot be calculated in one place. We thus propose a protocol to obtain optimal values step by step by communicating between the cloud and the edges. $cost_t()$ is a total cost function and the total optimization is defined to minimize this cost function under all the constraints. k is a parameter for properly balancing the processing of the cloud and of the edge. Although, here, a total cost is assumed to be a linear equation of $cost_c()$ and $cost_e()$, it may be a higher order equation depending on the system.

3.4 Protocol

Figure.6 shows the step by step protocol to obtain total optimized values under our proposed architecture.



Cloud
edge
time
t0
t1
t2
t3
MIN(cost_c(v_s(t0), v_c))
under constraint_c(v_s, v_c)
v_s(t0)
MIN(cost_e(v_s(t0), v_e))
under constraint_e(v_s0, v_e)
v_s(t1)
MIN(cost_c(v_s(t1), v_c))
under constraint_c(v_s1 , v_c)
v_s(t2)
MIN(cost_e(v_s(t2), v_e))
under constraint_e(v_s2, v_e)
v_s(t3)
Communicate with shared variable v_s as an intermediary

Figure 6: Protocol for Optimizing Between Cloud and Edges

At time $t0$, the cloud subtask calculates the global optimized values for $\boldsymbol{v_s}$ and $\boldsymbol{v_c}$ under $constraints_c(\boldsymbol{v_s}, \boldsymbol{v_c})$. After this step, the cloud subtask sends the result of $\boldsymbol{v_s}$ to the edges subtask. At time $t1$, the edge subtasks calculate the local optimized value for $\boldsymbol{v_s}$ and $\boldsymbol{v_e}$ under $constraints_e(\boldsymbol{v_s}, \boldsymbol{v_e})$. Then, it sends back new values for $\boldsymbol{v_s}$.

By advancing this communication step by step, the total optimization is realized with the common shared variables $\boldsymbol{v_s}$ as intermediaries. In an actual application, depending on the characteristics of the problem, the frequency of communication and the time required to complete the communication will be adjusted.

The process of obtaining optimal value depends on the application. In many applications, it is not necessary to precisely calculate the mathematical minimum value, and an appropriate value within an acceptable range is required within the required time.

In the case of multiple edges, the shared variable required for the optimization of the self-edge may be different for each edge, and optimization is performed through only the necessary shared variables. If different values are returned to the same shared variable between reply from multiple edges, the difference will be adjusted on the cloud side. The adjustment method depends on the application.

4 APPLICATION TO ENERGY MANAGEMENT SYETEM

In this section, we discuss an application of the flexible balanced IoT-EC architecture in energy management system.

In recent years, the electric power DR system that stabilizes the supply and price of electric power is progressing by reducing customers' electric power consumption when the power supply becomes limited and the electricity cost rises [29]. The basic structure of the DR system is shown in Figure.7. In the DR, a resource aggregator (RA) predicts the power demand of customers considering the weather forecast for the next day and the past power consumption records, and reduces the customers' power consumption so that the supply of power and demand match. On the other hand, the customers obtain incentive according to the ratio of the actual reduction amount to the reduction amount in their contracts. When customers join the DR system, they decide the contract reduction amount, which is the amount of electricity that can be reduced when required.

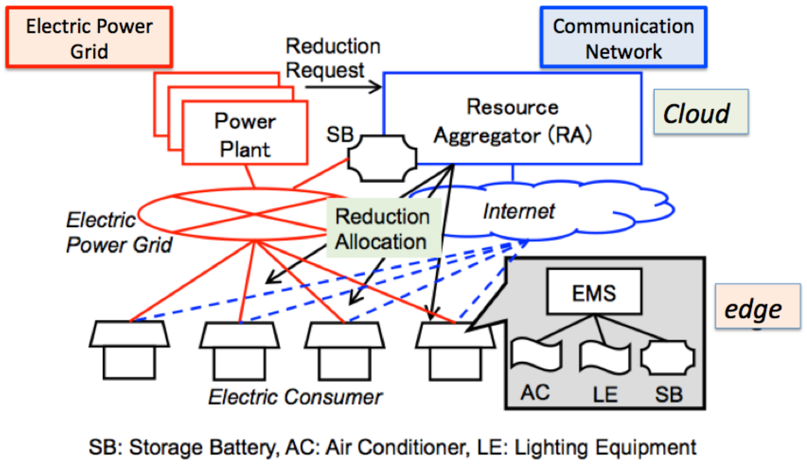


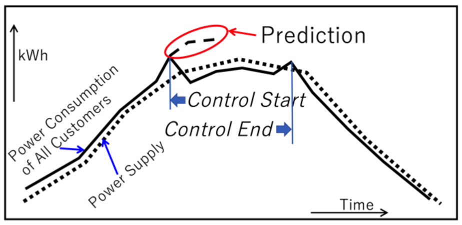
Figure 7: Electric Power Demand Response System

However, because of the introduction of photovoltaic power generation and storage batteries (SB in Figure.7) to customers, it has become difficult to predict their power consumption [30]. Further, the current DR takes into consideration only the maintenance of supply and demand balance of electric power (global optimization of electric power system), and does not consider comfort and energy conservation (local optimization of energy consumption) of each customer [9]. The current DR expects customers' voluntary demand adjustment by changing prices [31, 32]. In the future, in order to disseminate DR, providing the Advanced DR that considers both the global optimization of power system and the local optimization of energy consumption of customers will be required.

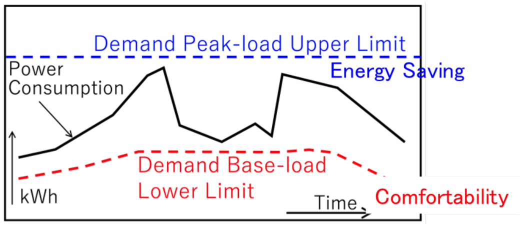
Before discussing the Advanced DR system, we explain the general behavior of the current DR system. The electric power company predicts the amount of electricity consumption from past data and makes plans for the electric power supply. If the actual power consumption greatly deviates from this supply amount, it leads to accidents such as power outage. Hence, some measures need to be taken. In the case of DR system, when the power consumption is expected to exceed the supply amount, the contract customer is requested to reduce their power consumption (Figure.8(a)). The request is sent from the power company to the RA, and the RA distributes the reduction amount to each customer.

At present, in the current DR system, the customer's needs are not taken into consideration, and the reduction amount is determined based on the contract. Each customer has a different contract reduction amount.

For the customer, there are two thresholds: the maximum power amount (demand peak-load upper limit) and the minimum power amount (demand base-load lower limit) in Figure.8(b). When power consumption exceeds the demand peak-load upper limit, the basic fee of the electricity charge usually increases for the following one year. The customer thus has to control the consumption so that it does not exceed the maximum thresholds. The demand base-load lower limit is the minimum amount of electric power required by the customer. When the electricity supply is below this value, facilities such as the air conditioner and lights will not function and the customer's comfort will be



(a) Global Optimization by RA

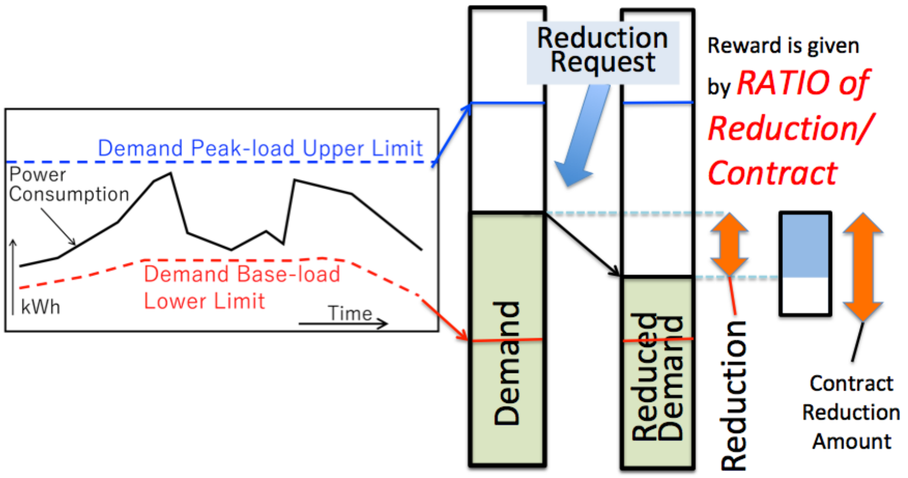


(b) Local Optimization by Customer's EMS

Figure 8: Optimization of Advanced DR System

impaired.

In the DR system, customers receive compensation according to the reduced amount ratio of the actual reduction amount and the contract reduction amount. If other conditions are satisfied, increasing the required reduction rate will increase the customer's reward (Figure.9).



Reduction Request
Reward is given by RATIO of Reduction/ Contract
Demand Peak-load Upper Limit
Power Consumption
Demand Base-load Lower Limit
kWh
Time
Demand
Reduced Demand
Reduction
Contract Reduction Amount

Figure 9: Control and Contract of DR system

4.1 Formulation of Advanced DR Optimization

In advanced DR, requirements and constraints of the global optimization to be achieved by the cloud service of RA and the local optimization to be achieved by an edge service of the customer's energy management system (EMS) should be as follows.

In a real environment, multiple RAs provide systems with different policies, and the customer selects the most suitable RA and agrees on a contract with that RA. In this paper, we assume and discuss the most common RA policy selected by customers. Moreover, for simplicity, we assume every edge has the same cost function.

4.1.1 Global Optimization by RA

The global optimization by RA is to minimize the cost function $Cost_c()$.

$$\underset{\boldsymbol{v_s},\boldsymbol{v_c}}{\text{MIN}}\, Cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}) \text{ under } constraints_c(\boldsymbol{v_s}, \boldsymbol{v_c}) \tag{7}$$

RA monitors power supply and electric power demand in real time. When electric power demand is expected to exceed electricity supply, the RA requests electricity reduction from the customers via the Internet. Figure.8(a) shows the global optimization of power supply and demand power adjustment made by the RA. The amount of requested reduction for each customer shall be allocated fairly to customers.

The cost function of the global optimization by RA can be expressed as Equation 8.

$$\begin{aligned}
&Cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(\frac{C_k(t)}{R_k} - \overline{CR(t)}\right)^2} \\
&\overline{CR(t)} = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{C_k(t)}{R_k}\right) \\
&\sum_{k=1}^{n} C_k(t) = \sum_{k=1}^{n} D_k(t) - S(t) \\
&\boldsymbol{v_s} = [(D_1(t), C_1(t)), (D_2(t), C_2(t)), \cdots (D_n(t), C_n(t))] \\
&\boldsymbol{v_c} = [S(t)]
\end{aligned} \tag{8}$$

Where $S(t)$ is the power supply amount at time t, $D_k(t)$ is the power demand of the customer k at time t, $C_k(t)$ is the reduction amount allocate to the customer k at time t, R_k is the contract reduction amount of the customer k, and n is the number of customers.

Equation 8 means that the cost function is the standard deviation of the ratio of the power reduction amount allocated to each customer to the contract power reduction amount. This is based on the idea that reduction allocation to customers is fair if the incentive percentage remains the same when possible. Constraint conditions in this global optimization problem are as follows.

$$C_k(t) \geq 0, D_k(t) \geq 0, R_k \geq 0, S(t) \geq 0 \tag{9}$$

To balance the supply and demand, it may be possible to utilize storage batteries (SB in Figure.7) installed in RA and at the customer's place. In this case, a mechanism to control the charging and discharging of the storage battery effectively is required [30]. However, this is not taken into consideration in this discussion. As the reliability of the electric power grid is very high, data losses are not considered in this example.

4.1.2 Local Optimization by Customer's EMS

The customer's EMS controls equipment in edge domain to maintain comfort and positive energy savings. Specifically, to maintain comfort, even when a request to reduce power consumption comes from the RA, electricity supply is controlled so that it does not fall below the minimum power consumption (demand base-load lower limits). Further, to maintain energy saving, electricity supply is controlled so that it does not exceed the predetermined power consumption upper limit (demand peak-load upper limits). Fig.8 (b) shows the local optimization to maintain comfort and energy saving by costumer's EMS.

The objective function of the local optimization by customer's EMS can be expressed as Equation 10.

$$\boldsymbol{v_s} = [(D_1(t), C_1(t)), (D_2(t), C_2(t)), \cdots (D_n(t), C_n(t))]$$

$$\boldsymbol{v_e} = [B_1(t), B_2(t), \cdots B_n(t)]$$

$$Cost_e(v_s, v_e) = \sum_{k=1}^{n} \left(\left(1 - \frac{C_k(t)}{R_k}\right) + S\left(\frac{B_k(t) + C_k(t) - D_k(t)}{R_k}\right)\right) \tag{10}$$

$$\begin{aligned} S(x) \quad &= x \quad \text{when } x > 0 \\ &= 0 \quad \text{when } x \leq 0 \end{aligned}$$

where $B_k(t)$ is the demand base-load lower limits to maintain comfort for the costumer at time t.

The first term of Equation 10 indicates the ratio of the allocated reduction amount to the contract power reduction amount. That is, when the customer can receive maximum incentive, this term becomes zero. The second term of Equation 10 indicates the cost of comfort. This is, when comfort is maintained, this term is zero. However, when the demand electric power falls below the demand base-load lower limit, we define the ratio of the excess amount to the contract reduction amount as the cost of comfort.

Constraint conditions in this local optimization problem are as follows.

$$\begin{aligned} &R_k - C_k(t) \geq 0 \\ &D_k(t) - B_k(t) - C_k(t) \geq 0 \\ &B_k(t) \geq 0 \end{aligned} \tag{11}$$

Where the electric power demand is significantly lower than the electricity supply, it may be possible to request the customer to increase the electric power demand by charging the storage battery (SB in Figure. 7). In this case, to maintain energy saving for the customer, it is necessary to add constraint conditions to prevent the electricity demand from exceeding the demand peak-load upper limits. However, this is not taken into consideration in this discussion.

4.1.3 Total Optimization

As shown in Equation 8 and Equation 10, both global optimization and individual optimization in the advanced DR system are time-series optimization problems. Furthermore, coordination between RA (cloud) and EMS (edge) is required so that each objective function can be minimized in a balanced manner. Here, assuming that the weights of optimization of the cloud and edge are equal ($k = 1$), the overall cost function of this system is presented as Equation 12.

$$Cost_t(\boldsymbol{v_s}, \boldsymbol{v_c}, \boldsymbol{v_e}) = Cost_c(\boldsymbol{v_s}, \boldsymbol{v_c}) + \sum_{\text{All edges}} Cost_e(\boldsymbol{v_s}, \boldsymbol{v_e}) \tag{12}$$

4.2 System Configuration

Figure.10 shows a system configuration diagram of the advanced DR system with the storage batteries based on the proposed flexible IoT-EC architecture. In the figure, the RA gathers power energy data from the customer's EMS once every half hour. In the case of limited power supply, the RA calculates the amount of reduction request to be allocated to each customer to balance power supply and demand based on Equation 8. On the other hand, the customer's EMS adjusts the electric power demand based on Equation 10 while maintaining the comfort and energy saving. The RA and EMS exchange $\boldsymbol{v_s}$ to balance global optimization of RA and local optimization of EMS.

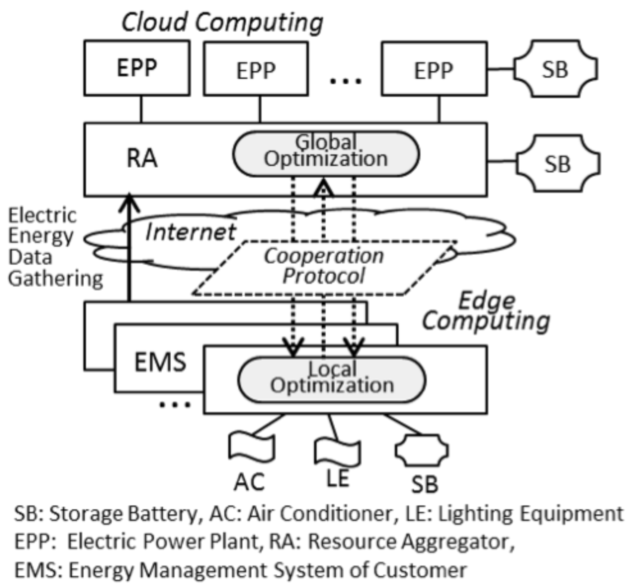

Cloud Computing
EPP
EPP
EPP
SB
RA
Global Optimization
SB
Electric Energy Data Gathering
Internet
Cooperation Protocol
Edge Computing
EMS
Local Optimization
AC
LE
SB
SB: Storage Battery, AC: Air Conditioner, LE: Lighting Equipment
EPP: Electric Power Plant, RA: Resource Aggregator,
EMS: Energy Management System of Customer

Figure 10: System Configuration of Advanced DR System

5 EVALUATION AND DISCUSSION

In this section, the effect of the proposed method is shown using two phase simulations. As previously explained, there are several cases where the cloud and the edges are in a trade-off relationship. The superiority of the proposed method is evident when the condition of the trade-off is severe. The first simulation is performed under three cases from a case with severe condition, that is, when the lower limit is high, to a case with relaxed condition, that is, when there is no lower limit. This simulation demonstrates the detailed behavior of the proposed system. To easy comprehension, the number of customers is kept at three. The second simulation is to confirm that the proposed method works well with multiple customers.

5.1 First Phase Simulation

Here, we describe a simulation example when the number of customers is three (C1, C2, and C3). The total supply is 600 kWh and the sum of demand for each customer is 900 kWh. The RA has to reduce 300 kWh and dispatch the reduced amount to each customer (Figure 11).

Simulation is conducted in three cases, 1) High Lower Limit, 2) Moderate Lower Limit, and 3) No Lower Limit (Figure. 12). Each customer has a different contract reduction amount, as shown in Table 1.

First, we describe the details of case 1). After that, the summary of cases 2) and 3) are shown.

5.1.1 High Lower Limit Case

As the lower limits in this case are the highest among the three cases, this scenario is the most severe. The RA tries to reduce $Cost_c()$ in Equation 8 and the results are requested from each customer, as shown in Figure. 13.

For customers C1 and C3, this request is acceptable. However, for customer C2, this request (reducing 177 kWh) results in the lack of demand.

$$300kWh - 177kWh = 123kWh < 200kWh (= \text{Contract Reduction Amount for C2}) \qquad (13)$$

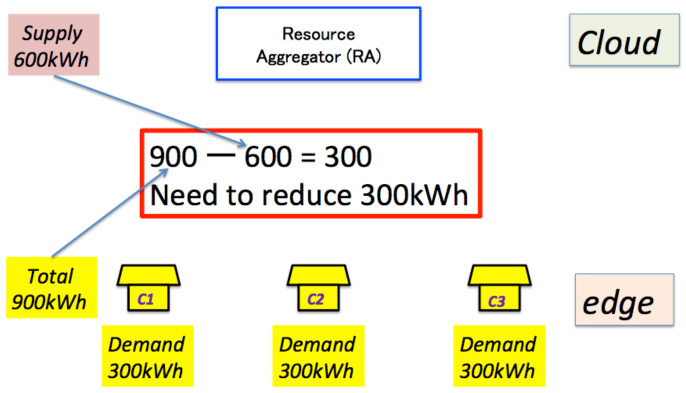


Figure 11: Simulation

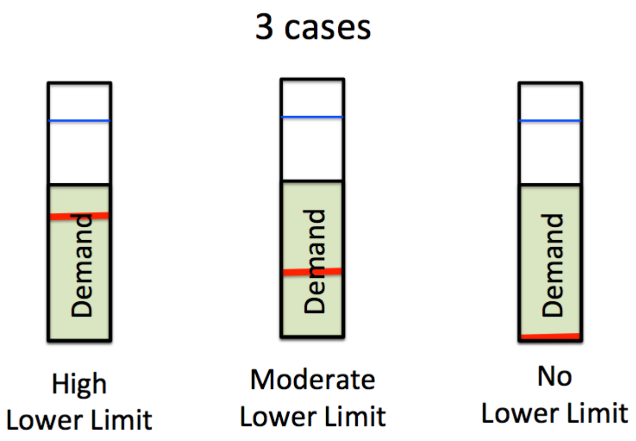


Figure 12: Simulation cases

Table 1: Simulation Conditions

		C1	C2	C3	Sum
$D_k(t_j)$ (= Demand) [kWh]		300	300	300	900
R_k (= Contract Reduction Amount) [kWh]		150	290	50	490
$B_k(t_j)$	High	100	200	250	460
(=Lower Limit)	Moderate	50	150	100	300
[kWh]	No	0	0	0	0

$D_k(t_j)$: power consumption of customer k at time t_j
R_k : contract reduction amount of customer k
$B_k(t_j)$: demand base-load lower limit of customer k at time t_j

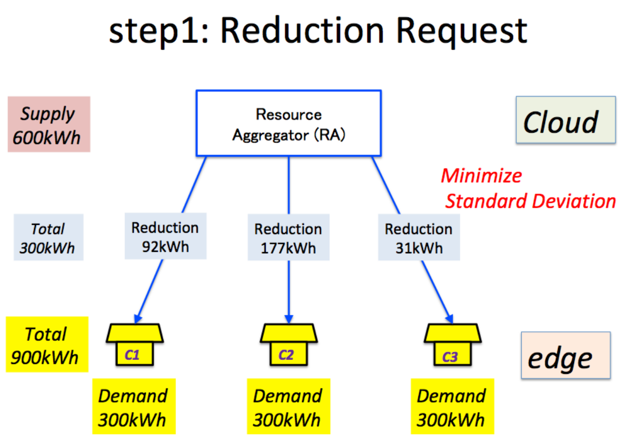


Figure 13: Step1: Reduction Request in case1

Thus, C1 and C3 reply with acceptance of the request, but C2 replies that only 100 kWh reduction is acceptable (Figure.14).

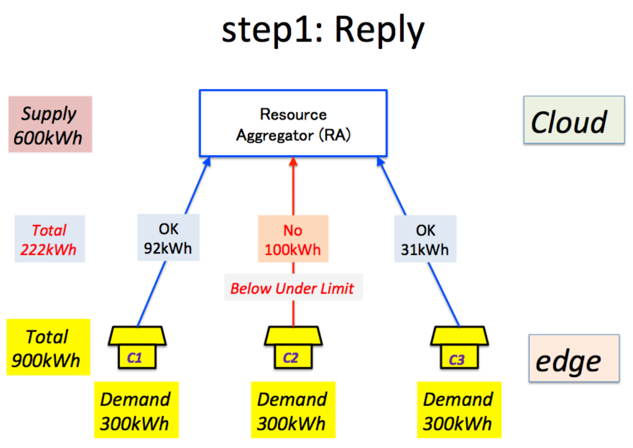


Figure 14: Step1: Reduction Reply in case1

The RA recalculates the reduction request for C1 and C3 based on the condition C2 for the reduction amount to be 100 kWh. Then, the RA resends the reduction request (Figure.15).

Here, all customers accept the request and reply, as shown in Figure.16.

Table.2 shows the result of cost calculation. In step 1 of the optimization, the cost of global optimization by the RA becomes zero. However, as the amount of reduction of customer C2 was lower than the demand base-load lower limits, the total cost of the optimization becomes 1.17. As a result of reflecting the constraint condition of customer C2 in step 2, although the cost of the global optimization by the RA increased to 0.15, the total cost improved to 0.96. The ratio of improvement is 33%.

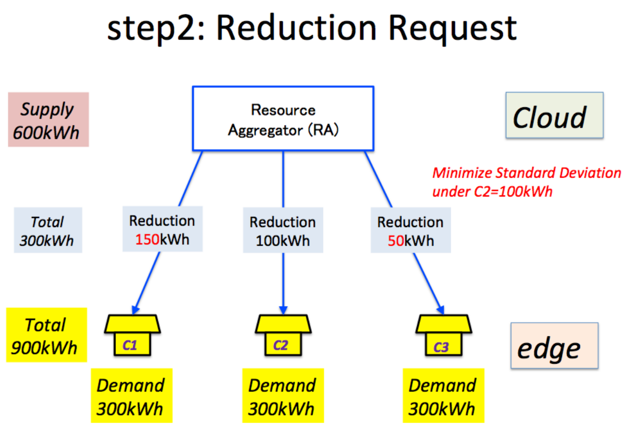


Figure 15: Step2: Reduction Request in case1

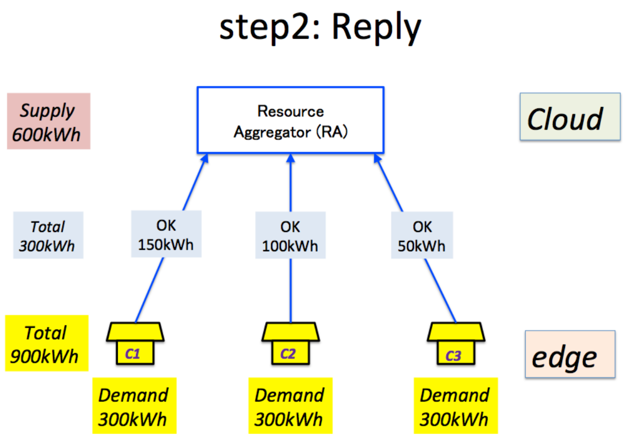


Figure 16: Step2: Reduction Reply in case1

Table 2: Results of Optimization for case1

		C1	C2	C3	RA	$Cost_t$
Step 1	$C_k(t_j)$ [kWh]	92	177	31	300	-
	$Cost$	0.39	0.66	0.39	0.0	1.17
		1.43				
Step 2	$C_k(t_j)$ [kWh]	150	100	50	300	-
	$Cost$	0.0	0.65	0.0	0.31	0.96
		0.65				

$C_k(t_j)$: reduction allocation amount to customer k at time t_j

Figure.17 shows the communication between the cloud and the edges in case 1). In order to make the figure clearer, every three edges are written as one edge. In fact, only the information on that edge is exchanged for each three edges, as shown in from Figure.13 to Figure.16.

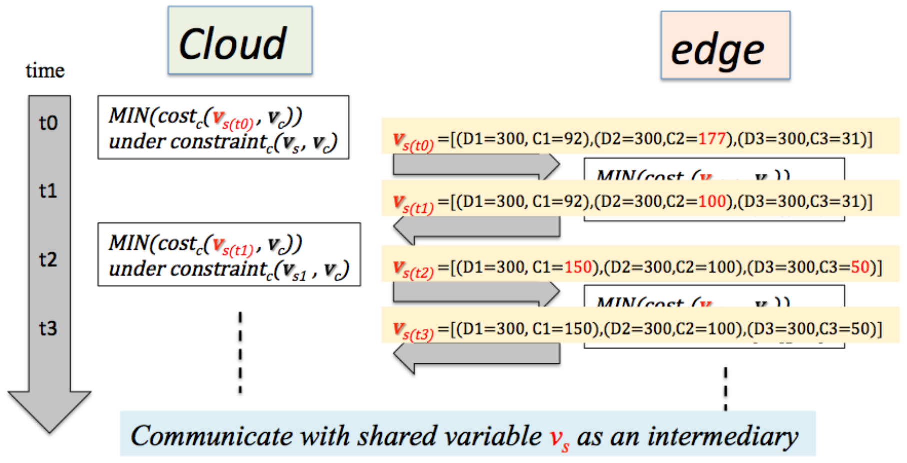

Cloud
edge
time
t0
t1
t2
t3
MIN(cost_c(v_s(t0), v_c)) under constraint_c(v_s, v_c)
MIN(cost_c(v_s(t1), v_c)) under constraint_c(v_s1, v_c)
v_s(t0) =[(D1=300, C1=92),(D2=300,C2=177),(D3=300,C3=31)]
v_s(t1) =[(D1=300, C1=92),(D2=300,C2=100),(D3=300,C3=31)]
v_s(t2) =[(D1=300, C1=150),(D2=300,C2=100),(D3=300,C3=50)]
v_s(t3) =[(D1=300, C1=150),(D2=300,C2=100),(D3=300,C3=50)]
Communicate with shared variable v_s as an intermediary

Figure 17: Protocol Simulation for case1

5.1.2 Other Cases

In the other 2 cases (Moderate Lower Limit, No Lower Limit), the final cost values are shown in Table 3 and Table 4.

In case 2), the total cost for step 1 is 1.26 and that for step 2 is 1.10. The ratio of improvement is 13%.

In case 3), the request of step 1 is accepted by all the edges and there is no need to proceed to step 2).

Figure.18 shows the results for all three cases.

Table 3: Results of Optimization for case2 (Moderate Lower Limit)

		C1	C2	C3	RA	$Cost_t$
Step 1	$C_k(t_j)$ [kWh]	92	177	31	300	-
	$Cost$	0.39	0.48	0.39	0.00	1.26
			1.26			
Step 2	$C_k(t_j)$ [kWh]	112	100	38	300	-
	$Cost$	0.25	0.48	0.25	0.12	1.10
			0.98			

Table 4: Results of Optimization for case 3) (No Lower Limit)

		C1	C2	C3	RA	$Cost_t$
Step 1	$C_k(t_j)$ [kWh]	92	177	31	300	-
	$Cost$	0.39	0.39	0.39	0.00	1.16
			1.16			

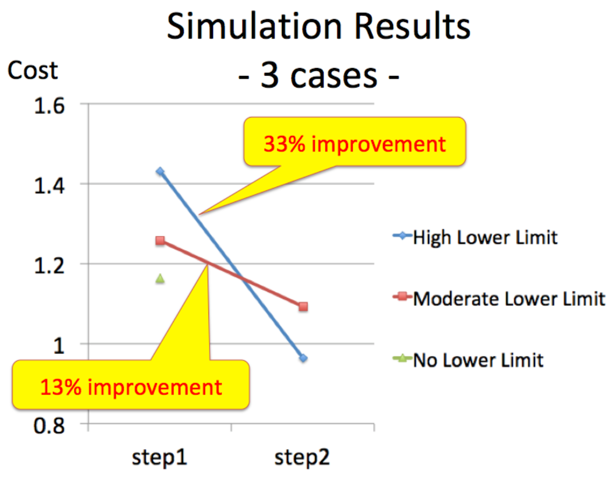


Figure 18: Simulation Results for All Cases

5.2 Second Phase Simulation

In the second phase simulation, we executed the simulation with a more reasonable number of customers and used the same method as previously described. The number of customers (N) changed from 10 to 10,000. In the case of Japan, the target customers of DR are estimated as approximately 700,000 [33]. Assuming that there are 10 electric power companies, the contract rate is 10%, and there are 10 aggregators in the same area. It is considered that there are approximately thousands of customers per aggregator. The current demand (D_k) is 300 kWh for all customers. The contract reduction amount (R_k) and the demand base-load lower limit (B_k) are set randomly between 0 kWh and 300 kWh. Supply power will be reduced to 200 kWh. The simulation used the same procedures as previously described. We measured the number of steps to obtain the optimum value and the ratio of reduction of the cost value. Depending on the value of the randomly set parameter, some cases were unable to obtain the optimum value. Such cases were removed from the simulation results. The simulation was executed 1000 times and the resulting average value is shown.

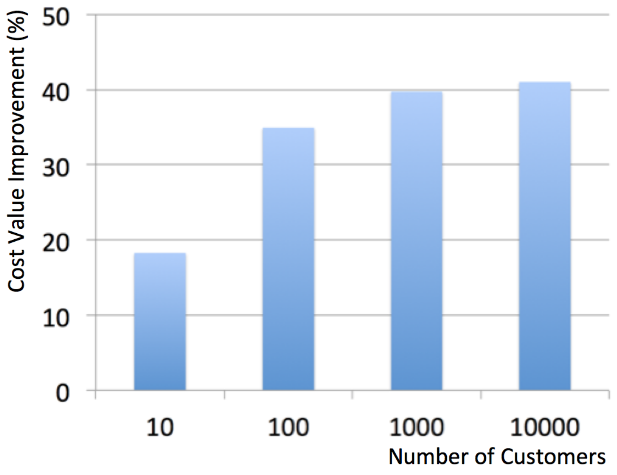


Figure 19: Cost Value Improvement in the Second Phase Simulation

The results are shown in Figure.19 and Figure.20. As seen in Figure.19, when the number of

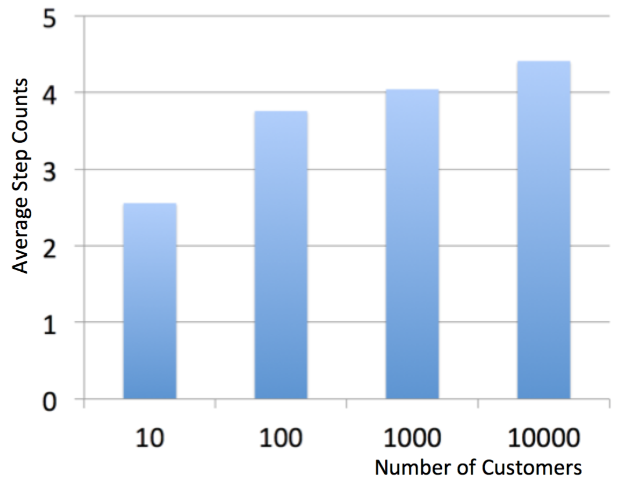


Figure 20: Average Step Counts in the Second Phase Simulation

customers increases, the degree of improvement of the cost function also increases. The average improvement ratio is approximately 40% when the number of customers is 1000 to 10,000. From Figure.20, we see the number of steps required to obtain the optimum value increases as the number of customers increases. It is approximately four to five steps with the number of customers being 10,000.

5.3 Discussion

As shown in the above simulation results, in the conventional DR system that does not consider the comfort and energy saving, although the cost of optimization of the cloud is 0, the cost of the edge increases (Step 1 in Table2, Table3). On the other hand, in the advanced DR systems, based on our proposed architecture, it is possible to balance the optimization of the cloud and the edge (Step 1 in Table2, Table3). The total cost is improved in the advanced DR system by 33% for case 1) and by 13% for case 2) (Figure 18) in the first phase simulation.

In the second phase simulation, we can identify improved performance number such as 40% or more with 1,000 or 10,000 customers. The number of steps required to achieve the optimal result is not significant, and we believe our proposed method can be applied to the actual advanced DR system.

In the advanced DR system shown above, we assumed that the cloud task (RA) is to maintain the electricity supply and the demand balance, and the edge task (EMS) is to maintain comfort and energy saving. Depending on the environment of the application, it may be necessary to optimize the roles of the cloud and the edge. In that case, it can be combined with the environment adaptive IoT architecture [6, 7].

Further, in the above simulation, the temporal fluctuation of electricity supply and demand is not considered. In fact, variables of the cost function and the constraint condition change with time and therefore need to be treated as a combination problem of the time series optimization. Moreover, as the number of customers increases, the optimization may not converge. Therefore, when applied to a real system, research on a more efficient distributed optimization algorithm is necessary. In addition, if there are storage batteries available with customers (SB in Figure 7 and Figure 10), it may be considered an effective method of distributing reproduced power autonomously among customer's EMSs without involving the RA of the cloud. For this purpose, we need research to solve Problem 3 described in Section 2.

Regarding the cost function for edges, we assume all the edges have the same cost function in this simulation. In our future work, we intend to evaluate more general cases in which each edge has a different cost function. We also aim to evaluate our architecture on other applications with mobile devices such as ITS systems.

6 CONCLUSION

In this paper, we proposed a flexible IoT-EC architecture to balance a global optimization in the cloud and a local optimization in the edge. We then showed its effectiveness by applying the proposed architecture to the electric power demand response system. As our future work, we intend to study the general formulation of the distributed optimization problem by clouds and edges and cooperative control among edges. For the application to electric power systems, we will study the distributed optimization between the RA and customers with storage batteries (SB in Figure.7 and Figure.10).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communication Surveys & Tutorials*, 17(4):2347–2376, June 2015.

[2] R. Buyya and A. V. Dastjerdi. *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, May 2016. ISBN13: 978-0128053959.

[3] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang, and W Liu. Study and application on the architecture and key technologies for IOT. In *Proceedings of IEEE International Conference on Multimedia Technology (ICMT)*, pages 747–751, July 2011.

[4] M. Abdelshkour. IoT, from Cloud to Fog Computing, Mar. 2015. `http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing` (accessed 18 Jun. 2017).

[5] P. G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, October 2015.

[6] N. Shiratori, S. Kitagami, T. Suganuma, K. Sugawara, and K. Shimamoto. Latest Developments of IoT Architecture. *The Journal of the Institute of Electronics, Information and Communication Engineers*, 100(3):214–221, March 2017. (in Japanese).

[7] T. Suganuma, T. Uchibayashi, S. Kitagami, K. Sugahara, and N. Shiratori. Proposal of An Environment Adaptive Architecture for Flexible IoT. *IEICE technical report*, 116(231):13–18, September 2016. (in Japanese).

[8] S. Kitagami, V. T. Thanh, D. H. Bac, Y. Urano, Y.Miyanishi, and N.Shiratori. Proposal of a Distributed Cooperative IoT System for Flood Disaster Prevention and its Field Trial Evaluation. *International Journal of Internet of Things*, 5(1):9–16, April 2016.

[9] S. Kitagami, M. Yamamoto, M. Imamura, H. Kambe, H. Koizumi, and T. Suganuma. An M2M Data Analysis Service System based on Open Source Software Environment. *The transactions of IEEJ. C*, 133(8):1521–1528, August 2013. (in Japanese).

[10] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah. Edge computing for the internet of things. *IEEE Network*, 32(1):6–7, Jan 2018.

[11] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato. Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control. *IEEE Transactions on Computers*, 66(5):810–819, May 2017.

[12] J. Ren, H. Guo, C. Xu, and Y. Zhang. Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing. *IEEE Network*, 31(5):96–105, 2017.

[13] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, and Y. Liu. A Survey on Emerging Computing Paradigms for Big Data. *Chinese Journal of Electronics*, 26(1):1–12, 2017.

[14] J. Ren, Y. Zhang, K. Zhang, and X. Shen. Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Communications Magazine*, 53(3):98–105, March 2015.

[15] M. Satyanarayanan. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Computing Communication*, pages 19–23, Jan. 2015.

[16] H. Chang, A. Hari, S. Mukherjee, and T. V. Lakshman. Bringing the cloud to the edge. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 346–351, April 2014.

[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4):14–23, Oct 2009.

[18] L. Gkatzikis and I. Koutsopoulos. Migrate or not? exploiting dynamic task migration in mobile cloud computing systems. *IEEE Wireless Communications*, 20(3):24–32, June 2013.

[19] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root. Tactical Cloudlets: Moving Cloud Computing to the Edge. In *2014 IEEE Military Communications Conference*, pages 1440–1446, Oct 2014.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.

[21] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu. Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing. *IEEE Transactions on Computers*, 65(7):2325–2331, July 2016.

[22] M. Marjanović, A. Antonić, and I. P. Žarko. Edge Computing Architecture for Mobile Crowdsensing. *IEEE Access*, 6:10662–10674, 2018.

[23] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust. Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things. *IEEE Consumer Electronics Magazine*, 5(4):84–91, Oct 2016.

[24] Y. Sahni, J. Cao, S. Zhang, and L. Yang. Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things. *IEEE Access*, 5:16441–16458, 2017.

[25] Muhammad Habib ur Rehman, Prem Prakash Jayaraman, Saif ur Rehman Malik, Atta ur Rehman Khan, and Mohamed Medhat Gaber. RedEdge: A Novel Architecture for Big Data Processing in Mobile Edge Computing Environments. *Journal of Sensor and Actuator Networks*, 6(3), 2017.

[26] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC ’12, pages 13–16, New York, NY, USA, 2012. ACM.

[27] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog Computing: Focusing on Mobile Users at the Edge, Feb. 2015. `https://arxiv.org/abs/1502.01815` (accessed 30 Mar. 2018).

[28] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities. In *Proceedings of the ASE International Conference on BigData 2015*, pages 28:1–28:6, New York, NY, USA, 2015. ACM.

[29] P. Palensky and D. Dietrich. Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads. *IEEE Transactions on Industrial Informatics*, 7(3):381–388, August 2011.

[30] H. Asano. Integration of Demand-side Resources in Power System Operation. *The Institute of Electrical Engineers of Japan*, 135(11):766–771, November 2015. (in Japanese).

[31] Koichi Kobayashi. Probabilistic Model of Consumers for Demand Response. *Proceedings of the Japan Joint Automatic Control Conference*, 59:650–652, 2016. (in Japanese).

[32] Takayuki ITO, Shantanu CHAKRABORTY, Takanobu OTSUKA, Ryo KANAMORI, Keisuke HARA, Ito Takayuki, Chakraborty Shantanu, Otsuka Takanobu, Kanamori Ryo, and Hara Keisuke. A Survey of Multi-Agents Research That Supports Future Societal Systems(2) : Power Systems, and Wireless Sensor Networks(<Special Issue>Agent Technology). *Journal of Japanese Society for Artificial Intelligence*, 28(3):370–379, may 2013. (in Japanese).

[33] Trade Ministry of Economy and Industry. Recent trends in the smart meter, Mar. 2012. `http://www.meti.go.jp/committee/summary/0004668/011_03_00.pdf` (accessed 7 June 2018) (in Japanese).