A Multi-agent Based Flexible IoT Edge Computing Architecture and Application to ITS

Tadashi Ogino¹, Shinji Kitagami², and Norio Shiratori³
¹Meisei University, Tokyo, Japan
²Fukui University of Technology, Fukui, Japan
³Chuo University, Tokyo, Japan
Email: tadashi.ogino@meisei-u.ac.jp, kitagami@tau.cugat.net, norio@shiratori.riec.tohoku.ac.jp

Abstract—On a large-scale IoT system based on cloud computing, problems such as increase of network load, delay in response, invasion of privacy, are concerned in recent years. As a solution to this problem, edge computing has introduced to the IoT systems. However, if you migrate the cloud function to the edge too much, the collected data cannot be shared between IoT systems and this decreases the usefulness of the IoT system. In this paper, a multi-agent based flexible IoT edge computing architecture is proposed 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. Also, as its application example, a parking search system example based on the proposed edge computing system architecture is shown to show the effectiveness.

Index Terms—IoT, edge computing, cloud computing, flexible, multi-agent

I. INTRODUCTION

Recently 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]. Applications of IoT system are adopted in the industrial sector, the household sector and the social sector. In the industrial sector, the increasing sophistication of remote maintenance and supply chain management is expected. In the home sector, the increasing sophistication of health care and energy management is expected. In the social sector, effective applications of the IoT system for disaster prevention, such as monitoring and alarm for a flood, are anticipated. These conventional IoT systems are cloud-centric architecture. Therefore, problems such as increase of network load, delay of feedback response, invasion of privacy are pointed out in a large-scale IoT system [4]. In order to solve the problems, the concept of the Edge Computing (EC) has been introduced to the IoT architecture [4], [5]. In this Edge Computing, the data collection function, filtering function and feedback control function are implemented on the edge servers in the career base-stations or the IoT gateways close to sensors and actuators. The Edge Computing is effective to solve the communication traffic shortage and the delay of feedback control function. However, if you migrate the cloud function to the edge too much, the collected data cannot be shared between IoT systems and this decreases the usefulness of the IoT system [6]. Also, while EC is effective for local optimization in an edge domain, global optimization of multiple domains cannot be realized.

To solve these problems, a couple of researches have been executed. In the previous research [7], a method to cooperate analysis of the data distributed to clouds and edges in the electric power system. Other research [8, 9] proposed an environment adaptive IoT architecture to optimize the roles of clouds and edges. This paper extends these previous researches and propose a multi-agent based flexible IoT edge computing architecture to solve these problems of the conventional EC. Proposed IoT architecture balances global optimization by a cloud and local optimization by edges and to optimize the role of the cloud server and the edge servers dynamically by multi-agent technology. In chapter II, the background of the research is shown. In chapter III, a concept of flexible IoT edge computing and its architecture are explained. In the next chapter, as an application example, a parking lot search system is explained and the effectiveness of the proposed architecture is shown.

II. BACKGROUND OF THIS RESEARCH

A. Conventional Cloud-centric IoT Architecture

A variety of IoT architectures are proposed by standards bodies and researchers [1, 3]. A couple of architecture is based on a 3-layer IoT architecture shown in Fig. 1 (a). In the 3-layer IoT architecture, as functional decomposition depends on the architecture, this architecture is regarded as vertical integration IoT architecture. Other types of architecture are 5-layer IoT architecture. This architecture extracts the common function from 3-layer IoT architecture and adds business layer. Fig. 1 (b) is one example of 5-layer IoT architecture proposed by Al-Fuqaha called API-based 5-layer IoT architecture.

The service management layer of 5-layer IoT architecture supports the IoT common functions, such as data collection, data analysis, device management and service discovery. These common functions are provided as the horizontal integrated IoT platform on the cloud. As a result, you can get some merits such as the reduction of...
the development cost, the protocol independency and the easy reuse of collected data.

In the 5-layered IoT architecture, all the data is collected and analyzed on the cloud. Also, all the actuators in the object layer are controlled by the result on the cloud. This behavior produces some demerits [4].

1) Under a huge IoT system with a lot of sensors, collecting huge amount of data makes the communication traffic shortage and declines the service level.

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

3) Collecting all the data on the cloud makes security issues more severe.

Fig. 1. IoT Architecture: a) 3-layer IoT Architecture; b) 5-layer IoT Architecture

B. IoT Edge Computing

To solve the problems shown in the previous section, the concept of the Edge Computing (EC) is introduced to the IoT architecture [5], [6]. EC is to perform the processing at the place near the data origin or control targets. A famous example is Contents Delivery Network (CDN). CDN deploys the Internet contents near the clients to improve Web performance.

Fig. 2 shows the IoT architecture with EC system (IoT-EC). In this architecture, the data collection function, filtering function and feedback control function are implemented on the edge servers in the career base-stations or IoT gateways closer to the IoT devices.

C. Problems of IoT Edge Computing

IoT-EC is effective to solve the network traffic shortage and the delay of feedback control. However, there are a couple of problems in IoT-EC as 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 the edge servers. That is, you need a method to optimize the total IoT system 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 and it prevents global optimization based on the collected data. On the contrary, giving priority to global optimization in cloud hinders local optimization such as real-time control in edge. That is, when an edge computing to the IoT system is introduced, you need a mechanism to balance global optimization by a cloud and local optimization by edges.

Problem-3) In IoT edge computing, local optimization at an 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.

III. FLEXIBLE MULTI-AGENT BASED IoT EDGE COMPUTING

There are some interesting leading researches [6-9] to solve the problems of IoT Edge Computing mentioned in the previous section. To solve the problem-1) of IoT-EC, an environment adaptive IoT architecture was proposed in the researches [6], [7]. Fig. 3 shows the concept of the environment adaptive IoT architecture. In the figure, "Environment Adaptability" autonomously assigns how to allocate 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 behavior of the user collected by IoT and detailed information such as expression and gesture.

Fig. 3. Concept of environment adaptive IoT architecture

A. IoT Edge Computing

Here the basic concept of how total balancing mechanisms work in the proposed architecture is explained.

The balancing optimization functions are divided to both the cloud side and the edge side. Each optimization
subtask can only optimize its side because it doesn't have enough information to take care the other side. The cloud side subtask, i.e. global optimization subtask, can improve its performance of the cloud, but that doesn't improve the edge performance like shown in Fig. 4. For the edge subtask (local optimization subtask), the situation is reversed and it cannot just 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. Just improving the performance of one side may decrease the performance of the other side.

![Fig. 4. Balancing Cloud Performance and Edge Performance](image)

In this research, the balancing optimization mechanism with the collaboration of both the global subtask and the local subtask is proposed. The goal of this mechanism is to achieve the total optimization of the system like in Fig. 4.

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

B. Architecture

Fig. 5 shows the overall architecture of the proposed architecture. An application is divided into multiple subtasks. Subtasks are assigned to a cloud or edges according to its characteristics as agents. For example, a global optimization subtask is placed in the cloud since it needs to access all the summary information gathered at the cloud. A real-time actuator control subtask requires low latency communication so it will be assigned to the edge servers connected to the actuators.

Subtasks are supposed to use autonomous distributed multi-agency technology. When necessary, agents can move from the cloud to the edges, from the edges to the edges, etc. This is how the challenge of problem-1) is solved. As the details are described in [7], this paper don't mention the details of this issue anymore.

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 whole system to work properly. If all information that determines the behavior of the total system are gathered in the cloud, the optimization function and control function can be executed only in the cloud. In many cases, such information exists at system-wide dispersedly and it is difficult for just a single agent in the cloud to control the whole system. When such agents exist both in a cloud and edges, both agents should collaborate to get the whole system to balance properly even from the cloud's point of view and edge's point of view. In this paper, the details of this optimization mechanism are explained in a later section. In some cases, one edge agent needs to communicate with different edge agent in the vicinity. Or if necessary, the cloud may communicate with other clouds to collaborate some task. In the proposed system, these functions are realized as the cooperation between edges/clouds. With those mechanisms described above, the proposed system can overcome 3 challenged encountered by cloud edge system.

![Fig. 5. Multi-agent based architecture of flexible IoT edge computing](image)

C. Formulation

To balance the optimization of the total application, the cloud subtask and the edge subtasks should communicate and negotiate the detail 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

At first, you need to understand the most simple case when all the information in the system can be accessed by a single subtask. There are multiple variables $v_i$ to affect the system behavior. To optimize the system means to minimize the cost of the system under some constraints by changing these variables.

\[
\text{variables: } \mathbf{v} = [v_0, v_1, \cdots v_{N-1}]
\]

\[
\text{cost function: } cost(\mathbf{v})
\]

\[
\text{optimization: } \min(cost(\mathbf{v})) \text{ under constraints(\mathbf{v})}
\]

2) IoT edge computing case

In the case when an application is divided to the cloud and the edges, the variables, the cost functions and the constraints vary depend on the node.

\[
\text{variables: } \mathbf{v}_c = [v_{c0}, v_{c1}, \cdots v_{cK-1}] : \text{from cloud}
\]

\[
\mathbf{v}_e = [v_{e0}, v_{e1}, \cdots v_{eL-1}] : \text{from edges}
\]
$\mathbf{v}_s = [v_{s0}, v_{s1}, \ldots, v_{SM-1}];$ from both

cost function:

\[
\begin{align*}
\text{cost}_c(\mathbf{v}_s, \mathbf{v}_e) & \text{ for cloud} \\
\text{cost}_e(\mathbf{v}_s, \mathbf{v}_e) & \text{ for edge} \\
\text{cost}_e(\mathbf{v}_e, \mathbf{v}_c, \mathbf{v}_e) = & \text{cost}_e(\mathbf{v}_e, \mathbf{v}_c) + k \times \text{cost}_e(\mathbf{v}_s, \mathbf{v}_e) \\
& \text{ for both cloud and edge}
\end{align*}
\]

(2)

global optimization (cloud):

\[
\min(\text{cost}_c(\mathbf{v}_s, \mathbf{v}_e)) \text{ under constraints}_c(\mathbf{v}_s, \mathbf{v}_e)
\]

local optimization (edge):

\[
\min(\text{cost}_e(\mathbf{v}_s, \mathbf{v}_e)) \text{ under constraints}_e(\mathbf{v}_s, \mathbf{v}_e)
\]

(3)

The variables $v_i$ are classified to 3 types according to accessible nodes, from the cloud, from the edges, and from both. $v_i$ is a shared variable and used to affect both the cloud and the edge system. $v_c$ is a variable used to affect only the cloud behavior. $v_e$ is a variable used to affect only the edge behavior.

$\text{cost}_c(\cdot)$ and $\text{cost}_e(\cdot)$ are calculated only in the cloud or the edges respectively. As a result, the total optimization cannot be calculated in one place. A protocol is proposed to get optimal values step by step by communicating between the cloud and the edges. $\text{cost}_e(\cdot)$ is a total cost function and the total optimization is defined to minimize this cost function under all the constraint. $k$ is a parameter for properly balancing the processing of the cloud and the processing of the edge. Although, a total cost is assumed a linear equation of $\text{cost}(\cdot)$ and $\text{cost}_e(\cdot)$ here, it may be a higher order equation depending on the system.

D. Protocol

Fig. 6 shows the step by step protocol to get total optimized values under the proposed architecture.

At time $t_0$, the cloud subtask calculates the global optimized values for $\mathbf{v}_s$ and $\mathbf{v}_e$ under $\text{constraints}_s(\mathbf{v}_s, \mathbf{v}_e)$. After this step, the cloud subtask sends the result of $\mathbf{v}_s$ to the edges subtask. At time $t_1$, the edge subtasks calculate the local optimized value for $\mathbf{v}_s$ and $\mathbf{v}_e$ under $\text{constraints}_e(\mathbf{v}_s, \mathbf{v}_e)$. Then it sends back new values for $\mathbf{v}_s$.

By advancing this communication step by step, the total optimization is realized with the common shared variables $\mathbf{v}_s$ as intermediaries. In an actual application, depending on the characteristics of the problem, the frequency of communication and the timing of finishing the communication will be adjusted.

IV. APPLICATION TO ITS

In this section, the application of the proposed architecture to the ITS system, especially the parking lot search system, is explained.

A. ITS and Parking Lot Search System

ITS (Intelligent Transport Systems) is a system that receives and transmits information between people, roads and cars and solves various problems. The automobile itself is an information processing apparatus having a large number of sensors and can be regarded as an edge node in the IoT system. Facilities such as people and parking lots around them are also edges, and this is a large-scale IoT system with clouds that aggregate and process information from these edges [11], [12]. In this section, the evaluation is done by applying the proposed architecture to a relatively simple parking lot search system.

B. Formulation of Parking Lot Search System

Here, a parking lot search system is defined as a system for finding a parking lot where a parking space is available. The billing system, the parking support system, the tour system, etc. are not considered. The system finds only one parking slot for one car only once. In the parking lot search system, each car designates a preferable parking lot to park and checks whether parking space is available or not.

Each car aims to minimize the waiting time before parking. The parking lot side aim to minimize the empty parking space.

\[
\text{Cost}_{\text{carN}} = T_{W_{\text{carN}}} \\
\text{Cost}_{\text{parking}} = 1 - U_{\text{parkingM}}
\]

(7)

$T_{W_{\text{carN}}}$ is a waiting time for car N. $U_{\text{parkingM}}$ is a usage ratio for parking lot M. The total cost function for all the edges is as follows.

\[
\text{Cost}_e = \Sigma T_{W_{\text{carN}}} + k \times \Sigma \text{Cost}_{\text{parking}}
\]

(8)

$k$ is a parameter for adjusting the weight of the waiting time of the car and the vacancy rate of the parking lot when calculating the total cost function.

There are multiple cars and parking lots, both of which are edges. Normally, each car and each parking lot exchanges information on a one-to-one basis individually. So there is no function to control the overall optimization and the cost function of the edge can not be calculated on the edge side. Therefore, in the proposed system, the aggregator in the cloud is introduced. It mediates cars and parking lots and calculate the total cost function.

\[
\text{Cost}_e = \Sigma T_{W_{\text{carN}}} + k \times \Sigma \text{Cost}_{\text{parking}}
\]

(9)

C. System Configuration

When there is no overall optimization function, as shown in Fig. 7, cars and parking lots individually perform edge-to-edge communication to search available
parking spaces. This is called as a traditional edge-only system. In the proposed system, cars and parking lots communicate between the edge and the cloud via aggregators that manage overall optimization placed in the cloud as shown in Fig. 8.

D. Communication Protocol

When searching for an available parking space, each car sends a request for a desired parking lot to the aggregator (Fig. 9). The aggregator inquires the parking lot. When there is available space in the designated parking lot, the aggregator reply that information to the car. When there is no available space in the designated parking lot, the aggregator tries to find the different parking lot and reply the result to the car.

E. Simulation

The simulation is performed in a simple case with two parking lots (Fig. 10). The cars reach the target parking lot according to the exponential distribution. If there is no parking space, in the traditional edge-only system, the cars will wait until the parking lot becomes available. In the proposed cloud edge system, the aggregator checks the other parking lot. When it is vacant, the car moves and parks at the other park. At this time, the car needs to travel to the other park, and this travelling time is included in the waiting time.

1) Simulation case0

The number of parking space for parking lot 0 is 10 and 10 for parking lot 1. The average departure interval is 10 minutes for each parking lot. The average parking period is from 50 minutes to 100 minutes. The travelling time to change the parking lot is 5 minutes. The total simulation number of cars is 5,000.

The result waiting time is shown in Fig. 11. The waiting time for the cloud edge system is reduced by over 30% compared to the traditional edge-only system, even including the traveling time.

2) Simulation case1

The simulation results when the number of parking space for parking lot 1 is increased to 15 are shown in Fig. 13. In the conventional system, the waiting time of the parking lot 1 has decreased sharply, but the waiting time of the parking lot 0 is the same as before, so the average value is about half of the previous result. In the proposed system, waiting times for both parking lot have decreased significantly.
The utilization rate of the parking lot as a whole doesn't change so much (Fig. 14).

In the two cases above, the utilization rate of the entire parking lot itself does not change, but the waiting time has been reduced by over 30%. In the actual case, it is expected that as the waiting time decreases, the number of cars that can be accommodated increases and the total utilization rate will improve accordingly.

V. CONCLUSIONS

In this paper, a flexible IoT edge computing architecture is proposed, which is to balance a global optimization in the cloud and a local optimization in the edge. Also, applying the proposed architecture to the parking search system, its effectiveness is shown. Future works include the general formulation of distributed optimization problem by clouds and edges and cooperative control among edges. Other applications will be shown to show the universality of the proposed architecture.

REFERENCES


Shinji Kitagami received his PhD in Information Sciences from Tohoku University, Japan, in 2013. He is currently a Professor of Faculty of Environmental and Information Sciences at Fukui University of Technology, Japan. He received Best Paper Award from FA Foundation in 2016, etc. His main research interests include IoT edge computing, cloud security and energy management. He is a member of IEICE, IPSJ and IEICE.

Dr. Norio SHIRATORI is currently Professor of Chuo University, Tokyo and Professor Emeritus of Tohoku University, Sendai. He is the IEEE Life Fellow, fellows of the JFES (The Japan Foundation of Engineering Societies), the IPSJ (Information Processing Society of Japan), and the IEICE (The Institute of Electronics, Information and Communication Engineers). He was the president of the IPSJ (2009-2011), the Chair of the IEEE Sendai Section (2010-2011) and the Vice Chair of the IEEE Japan Council (2013-2014). He has received Science and Technology Award (Research Division)” by MEXT (Ministry of Education, Culture, Sports, Science and Technology-Japan) in 2009, IEICE Honorary Member, IPSJ Honorary Member and many others.