# Design and Implementation of Multi Agent Simulator for Resource Transparent Widely Distributed Computing Environment

Hiroki Kashiwazaki[1,a)] Yutaka Kikuchi[3] Ikuo Nakagawa[4]
Kazuma Nishiuchi[5] Mitsuhiro Osaki[5] Shunsuke Kikuchi[6] Hideki Takase[2]

**Abstract:** The authors seek to pioneer an innovative computing environment for the Beyond 5G era by using Elixir, a highly scalable and fault-tolerant functional language. First, we build a massively parallel distributed processing infrastructure transparent to various components of a wide-area distributed system. Next, we will develop ultra-low-power processing and communication middleware that actively utilizes the characteristics of SoC hardware as the execution environment for the IoT nodes that are the system components. In addition, we will develop a method to determine the optimal allocation of computational resources required according to the functions and processing of IoT applications. By co-creating these platforms and algorithms, we can drastically reduce the cost of system construction and bring DX to the manufacturing and primary industries. This paper describes a method for determining the optimal allocation of computational resources using multi-agent simulation.

## 1. Introduction

In IoT systems, there are processing elements such as generation and transmission of sensing data from many sensors, collection, processing, and storage of such data, and control of actuators and other devices. The entities that implement these processes are the edge devices, the network, and the cloud. The edge devices (or the gateways that connect the edge devices to the network) and the cloud are the ones that currently have the computational power. The edge devices (or gateways connecting the edge devices to the network) and the cloud currently have the computational power. First of all, both computational resources are at the extremes of computational power and communication latency, and there is no element with the characteristics of an intermediate computational resource. In addition, the system configuration architectures of both are based on entirely different concepts, so building a system that uses both together requires paying a high technical learning cost.

Furthermore, it is necessary to consider that the main focus of IoT system development and application will shift to industries far away from ICT in the future. In these industrial usage scenarios, the requirements for response performance, processing content, reliability, and cost will become more diverse. The current typical architecture consists of a combination of edge devices. The cloud is expected to have difficulty in absorbing the variety of functions, performance, and costs required in the future. The current typical architecture, which consists of a combination of cloud computing, will be unable to absorb the variety of functions, performance, and costs that will be required in the future. In addition, since users are not expected to be well versed in various ICT technologies, system integrators will have to do a great deal of work in defining the requirements and specifications of the services desired by users. Therefore, an environment that combines and designs various technological services in a shorter period is required to relatively reduce the time and cost to grasp and catch up with the diversity of technologies related to design and implementation. In addition, developers of industrial systems need a system architecture where the learning cost is sufficiently small for them to use ICT technologies.

On the other hand, the following technological elements have emerged to create a technical environment for solving these issues. A 5G standard allows non-carriers to deploy MEC as an intermediate component of the network. Heterogeneous SoC (heterogeneous SoC) with low cost and high performance can flexibly respond to various system sizes and performance requirements. A practical programming language and its processing system based on a new paradigm of functional programming with actor-model compliant concurrency. Communication middleware for wide-area dis-

[1] National Institute of Informatics
[2] The University of Tokyo
[3] Kochi University of Technology
[4] Osaka University
[5] Citynet, Inc.
[6] SAKURA internet, Inc.
[a)] reo_kashiwazaki@nii.ac.jp

tributed processing that can configure loosely coupled system architectures. Development of open source (software and hardware) and engineering communities

Elixir[*1], the cornerstone of our research and development, is a parallel processing-oriented functional programming language that has been developed in open source since 2012. Elixir is a parallel processing-oriented functional programming language designed as an open-source since 2012. It is suitable for building large, scalable, soft real-time systems that require high availability, mainly used in telephone exchanges, bank account management systems, and messaging servers. Elixir has the following features and advantages. Elixir has the following features and advantages: It uses an actor model, where each process communicates with other processes only through non-shared and asynchronous message exchange. Since all data is handled immutably, there is no need to perform exclusion control or synchronization processing at the application level, and concurrent and parallel processing performance can be easily achieved. Furthermore, since BEAMs[*2] running on multiple nodes can communicate with each other, it is also suitable for realizing distributed systems.

The BEAM runs as one of the processes managed by the OS kernel, and each Elixir process is allocated a unique memory block. As a result, the process model is very lightweight and robust, making it easy to achieve economies of scale based on computational resources. Elixir's language specification is based on Ruby and is designed to be simple in notation and concepts but easy to apply. Furthermore, elixir is a functional paradigm that is highly descriptive and easy to learn, resulting in high development productivity. Therefore, the authors are trying to create an IoT architecture that actively utilizes the advantages and functions of the communication infrastructure in the Beyond 5G (B5G) era and pioneer an innovative computing environment based on Elixir, an actor-based functional language.

Utilizing the Multi-access/Mobile Edge Computing technology, a component deployed in the network from edge devices to the cloud, we will research and develop a transparent parallel processing infrastructure that can flexibly allocate processing to the resource characteristics of these nodes. We will be able to design and configure systems consisting of multiple IoT nodes (hereafter referred to as IoT systems) without being aware of the characteristics of the computational resources of IoT nodes distributed on the network (computational power, storage capacity, communication latency, etc.). Achieve resource transparency for processing applications distributed across IoT nodes that can operate in parallel with a response time of within 1-2 ms for local networks and within 10-300 ms for global networks.

Research and develop technologies that enable Elixir processes to run efficiently on IoT nodes (edge devices, MECs, and clouds), IoT systems' building blocks. We will build an execution method for the Elixir process and its processing system that actively utilizes the characteristics of the SoC (System-on-a-Chip) hardware from the virtual machine and achieve a power efficiency improvement of more than ten times compared to the conventional method. In addition, we will develop communication middleware specialized for communication in the B5G interconnection network. We will also prove that it can realize the ultra-high-speed, large capacity, ultra-low latency, and ultra-multiple simultaneous connections, which are functions that B5G should have.

Research and develop an algorithm to determine the optimal resource allocation to demonstrate the quality and performance of applications running on IoT nodes (hereafter referred to as IoT apps). Depending on the rewards and priorities for the functional requirements of the IoT app, we will be able to allocate, deploy, and execute IoT nodes that can maximize quality and performance. Simulation evaluation will prove that our algorithm works theoretically in a system consisting of more than 100 IoT nodes. We will connect a total of six computer servers to the widely distributed platform Distcloud and prove that our algorithm works practically. We will develop applications for demonstration and evaluation using these research and development results. We will show that our approach can significantly reduce IoT system development and operation costs and duration. Prove that the results of this R&D can serve as an entry for social implementation and commercialization.

By engaging in these R&D activities, we will create an architecture that can maximize the functions and performance required to develop and operate IoT systems. In addition, we will provide a unified, wide-area distributed platform that is independent of the characteristics and types of computational resources and a development environment for its applications. System developers who use the results of this research and development will be able to program in an integrated manner using a functional paradigm without being aware of the characteristics of the various resources such as edge devices and MECs that are components of IoT systems. Each process of the developed IoT application will operate transparently on each IoT node distributed on the network by optimally allocating computational resources through the allocation decision algorithm. In addition, the functions of the Elixir processing system and communication middleware equipped as the execution environment of the IoT nodes automatically realize parallelization and ultra-low power consumption of the entire IoT system massively.

In other words, by utilizing the results of this research, the learning cost of system development can be reduced, and the cost and time of system development can be greatly shortened. This will promote the use of IoT technologies in "industries where ICT is not the core business," such as manufacturing and primary industries, and bring about digital transformation (DX) in these industries.

## 2. Related Works

For the evaluation of IoT applications in edge computing,

[*1] The Elixir programming language `https://elixir-lang.org/`
[*2] The Erlang BEAM Virtual Machine Specification `http://www.cs-lab.org/historical_beam_instruction_set.html`

Sumit Maheshwari et al[1]. For IoT applications, especially targeting smart cities, the work of Sharu Bansal et al[2]. In particular, the contribution of optimization methods in Robot Operating System (ROS) may benefit from the approach of Ziyue Jiang et al[3]. There have been various studies on data distribution services (DDS) in industrial automation for a relatively long time, and the approach of Jinsong Yang et al. is one of the relatively well-known ones[4]. Topic naming is one of the important issues in DDS, and Gunjae Yoon et al. have addressed this issue[5]. As for the research on providing scalable DDS using mobile devices, Lincoln David et al. have been working on it since relatively early days[6]. Xiaokang Zhou et al. worked early on (2015) as a distributed learning method in the next generation communication networks called B5G or 6G[7].

## 3. Methodology

This research and development will contribute to constructing an innovative computing environment for the B5G era and to the effective use of radio waves. In particular, in the R&D area that the author is in charge of, the IoT applications will be assigned to computational resources that satisfy the quality and characteristics (latency characteristics and processing performance) required by the IoT applications to be deployed and executed. Computational resources are priced according to their operational quality, and we introduce a mechanism that allows an IoT app to select multiple computational resources concerning these prices dynamically. In the following, we describe the details of our proposed method.

### 3.1 Optimal allocation algorithm

One of the characteristics of the Mobile Edge Computing environment (MEC) is that geographic conditions dominate it. For example, users in the vicinity of a given MEC will tend to use applications with short response latency for information available at that MEC. On the other hand, some users want to deploy a uniform algorithm in the MEC to provide a consistent application independent of geographical conditions.

MECs deployed at the gateways of IoT nodes temporarily store the information collected by the IoT nodes and provide this information for applications with short request-response latency. However, the Service Level Agreement (SLA) provided by the MEC is not necessarily the same for MECs in different geographic conditions. Especially in the case of Local5G networks, no clear SLA may be set, so the quality of communication between the IoT node and the MEC may not be sufficient to ensure the quality of communication due to the return on investment by the investor in the MEC. The same can be said for the MEC and Internet communication.

The MEC and the user who provides the line will be provided with the MEC currency obtained from the billing. The user who provides the MEC and the communication line obtains the MEC currency through billing. The user who provides the MEC and the line accepts the MEC currency through billing. In other words, the user who provides the resources that make up the MEC and is committed to maintaining the high virtual Service Level (vSL). That is compensated in the currency that is accepted in the MEC (virtual Edge-network Currency (vEC)) in the amount that the provided resources are used. The vEC can then be used to purchase information obtained from the MEC. On the other hand, users who do not offer resources for this MEC can buy the information obtained in this MEC by exchanging real-world currency for the currency in the network.

When such an incentive is set up, the excessive investment will be curbed based on economic principles even if excessive resource investment is made because vEC cannot be obtained if the resources are not used. On the other hand, if the resources of the MEC are sufficiently small concerning the demand to purchase the information obtained within the MEC, the quality of operation will be below the SLA, resulting in a low vEC. Therefore, it provides a positive incentive for resource investment.

We have already conducted agent simulations in which users are made to behave as agents in an environment where this billing algorithm is implemented. In this research and development, we will also implement this in a real-world wide-area distributed network, conduct a demonstration experiment, compare the results with those of the simulation, and tune the parameters.

### 3.2 Priority control and conflict resolution algorithms

Two problems have been identified in the preliminary demonstrations. The point is that all information acquisition requests are treated equivalently in FIFO in MEC. Therefore, it can be assumed that users want to make high-priority requests by paying a high price. On the other hand, if an algorithm with such a priority is introduced into the MEC, a malicious third party can generate a temporary concentration of high-priority information acquisition requests, thereby reducing the actual vSLs observed in that MEC. It is expected that a mutual monitoring mechanism will be introduced in inter-MECs that connect different MECs against such attacks.

Another problem is expected when the inter-MEC as mentioned above is introduced. When inter-MEC is introduced, it can be assumed that the exchange of vEC, the currency within the MEC, will be required between different MECs. Since the exchange rate between the real currency and the vEC is determined by the observed practical service level vSL, the exchange rate between MECs is likewise determined by the ratio of the vSL. When an MEC achieves a high vSL by rejecting requests for information acquisition from arbitrary users, that MEC can create a vEC with an unreasonably high rate. It is expected that a mechanism will be introduced to curb such self-induced inflation of the monetary value.

In this section, we will design and implement a MEC that provides a billing service with an additional priority flag,

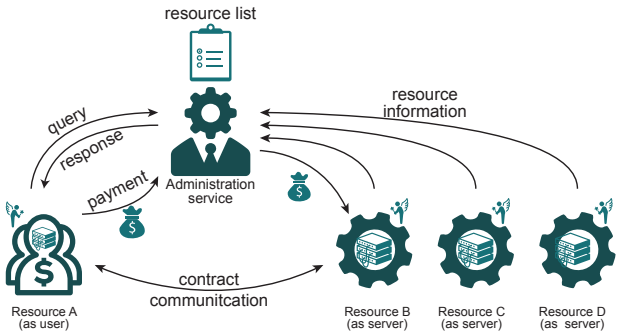


**Fig. 1** Relations among the resources and the administration service

a mutual monitoring mechanism in the inter-MEC, a service to determine the exchange rate of different vECs, and a mechanism to deter unfair rate inflation, and evaluate it through both simulation and feasibility studies.

### 3.3 Proposed Method

In MEC, computational resources are servers that provide computational resources and users that request computational resources for the programs they (or their owners) execute. Unlike cloud computing environments, where uniform computational resources are homogeneously managed by their owners, MEC comprises tens of thousands of types and hundreds of millions of total numbers. Let's not have any illusions that these devices can be centrally managed. Computational resources that meet the low latency requirement, one of the characteristics of edge computing environments, are geographically constrained and extremely limited in number. You should not, by mistake, advertise to the entire population and search for devices that match the criteria. The ad traffic will deplete your resources. My impression is that the ideal edge computing environment is lawless. Swarms are formed by devices communicating with each other and where there is no clear boundary between hives.

However, let's face reality and define a group of devices under the authority of an edge controller as a "domain" or a "village" environment. All resources (and their owners) are inconsistent, far from philanthropic, yet rational. While it would be my ideal to build a completely decentralized model, we will start with a centralized model and then asymptotically move to a decentralized environment. In other words, the ideal environment for this research is one where there is no centralized management service and no need for agents to be installed uniformly on all devices. Then build the model assuming that there is a management service and that agents are installed uniformly on all devices.

The agent installed on each resource measures the product of the amount of CPU, memory, storage space, and traffic demand used by a particular process on the OS that manages the resource and the time taken by the process and reports this to the management service. It also measures and reports the latency to all other resources in a domain. When a resource needs its own computational or network resources and those owned by other resources, this resource (the requester) sends a query request to the management service. This query includes the number of cores and operating frequency required to run the process and the maximum delay time needed to communicate with its resource. If there is no upper limit to the latency time, the cloud computing environment can be used instead of MEC, which provides limited computing and network resources. The management service that receives the query creates a resource list from the reports it receives from time to time and replies to the requestor with resources that match the constraints described in the query. This response may include multiple resources. After receiving the response, the requester selects one or more of these resources and requests the job to be executed via the management service. If the resource that receives the request accepts the job execution, the job will be handed over via the management service. After the job execution, the resource will report the information of the consumed computer resources and network resources. The management service bills the requestor for the amount of resources consumed (Figure 1).

The requesting resource does not decide which resources to use based on uniquely defined rules. It could be a rule to use the resource at the top of the list of resources received or use the resource with the lowest latency. Or the resources may have a history and prioritize specific resources based on their past performance, or vice versa, i.e., exclude specific resources found on their past performance. In other words, there are multiple ways to select resources. On the other hand, the same can be said for the resource that receives the request to execute a job. A resource can either accept or reject the request, offer the same number of cores as requested after receiving the job or offer only a limited number of cores to execute the job. It can also choose to oversubscribe to multiple job requests, or it can choose to provide computing resources exclusively (monopolistically) to other job requests. We have already discussed how such a job execution strategy provides an "impression" to the requester if the requester adopts a history-based selection method.

We do not know if the resource will adopt an "honest" strategy by providing such an economic incentive. This is partly because jobs are not uniform. A complex interplay of various parameters may sometimes lead to a local optimum solution and sometimes to an asymptote to the global optimum. A significant factor in this situation can be the "pricing" of computer and network resources. Depending on the pricing, some resources may be more conservative in using their computing and network resources, while others may be more liberal. We will try to use agent simulation to understand these situations quantitatively (Figure 2).

## 4. Evaluation

I wanted to talk about the design and implementation of multi-agent simulations, but I have no time. I started writing this manuscript only at 16:00 on February 1, 2022, the deadline for submitting this manuscript. I'll talk more

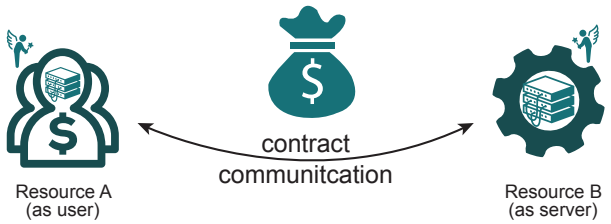


**Fig. 2** Economical Incentives among the resources

about it at the March meeting, and I'll start with a simple example of implementation in Python. Then, I will show the difficulties in empirically evaluating multi-agent simulations and discuss what approaches are effective in speeding up the computation.

## 5. Conclusion

The authors enumerate the problems of a resource-transparent hyper-distributed parallel environment using the Elixir functional language for the B5G environment in the future, when 5G is sufficiently widespread, and devise methods to solve them. This paper explains the requirements for optimal resource allocation, priority control, and conflict resolution algorithms, and an economically motivated approach that can solve these problems is described. The evaluation of the environment using multi-agent simulation shall be done in an oral presentation.

**Acknowledgments**

These research results were obtained from the commissioned research (04001) by National Institute of Information and Communications Technology (NICT), JAPAN. This work was also supported by JSPS KAKENHI Grant Number 19K20256.

### References

[1] Maheshwari, S., Raychaudhuri, D., Seskar, I. and Bronzino, F.: Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Applications, *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 286–299 (online), DOI: 10.1109/SEC.2018.00028 (2018).

[2] Bansal, S. and Kumar, D.: IoT Application Layer Protocols: Performance Analysis and Significance in Smart City, *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6 (online), DOI: 10.1109/ICCCNT45670.2019.8944807 (2019).

[3] Jiang, Z., Gong, Y., Zhai, J. and et al.: Message Passing Optimization in Robot Operating System, *Int J Parallel Prog.*, Vol. 48, pp. 119–136 (online), DOI: 10.1007/s10766-019-00647-w (2020).

[4] Yang, J., Sandström, K., Nolte, T. and Behnam, M.: Data Distribution Service for industrial automation, *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–8 (online), DOI: 10.1109/ETFA.2012.6489544 (2012).

[5] Yoon, G., Choi, J., Park, H. and Choi, H.: Topic naming service for DDS, *2016 International Conference on Information Networking (ICOIN)*, pp. 378–381 (online), DOI: 10.1109/ICOIN.2016.7427138 (2016).

[6] David, L., Vasconcelos, R., Alves, L. and et al: A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes, *Journal of Internet Services and Applications*, Vol. 4, No. 16 (online), DOI: 10.1186/1869-0238-4-16 (2013).

[7] Zhou, X., Liang, W., She, J., Yan, Z. and Wang, K. I.-K.: Two-Layer Federated Learning With Heterogeneous Model Aggregation for 6G Supported Internet of Vehicles, *IEEE Transactions on Vehicular Technology*, Vol. 70, No. 6, pp. 5308–5317 (online), DOI: 10.1109/TVT.2021.3077893 (2021).