J Inf Process Syst, Vol.17, No.5, pp.905~917, October 2021
https://doi.org/10.3745/JIPS.01.0080

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Range Segmentation of Dynamic Offloading (RSDO) Algorithm by Correlation for Edge Computing

Jieun Kang*, Svetlana Kim*, Jae-Ho Kim**, Nak-Myoung Sung***, and Yong-Ik Yoon*

Abstract

In recent years, edge computing technology consists of several Internet of Things (IoT) devices with embedded sensors that have improved significantly for monitoring, detection, and management in an environment where big data is commercialized. The main focus of edge computing is data optimization or task offloading due to data and task-intensive application development. However, existing offloading approaches do not consider correlations and associations between data and tasks involving edge computing. The extent of collaborative offloading segmented without considering the interaction between data and task can lead to data loss and delays when moving from edge to edge. This article proposes a range segmentation of dynamic offloading (RSDO) algorithm that isolates the offload range and collaborative edge node around the edge node function to address the offloading issue.The RSDO algorithm groups highly correlated data and tasks according to the cause of the overload and dynamically distributes offloading ranges according to the state of cooperating nodes. The segmentation improves the overall performance of edge nodes, balances edge computing, and solves data loss and average latency.

Keywords

Balancing, Collaboration Edge Computing, Context-Awareness, Data-Intensive Offloading, IoT, RSDO, Task-Intensive Offloading

1. Introduction

Advanced Internet of Things (IoT) devices collect real-time data and data stored in the cloud in large quantities [1-3], and they sufficiently perform various analysis techniques on their own. Edge computing improves response time and saves bandwidth by offloading to edge gateways (IoT devices) that can perform big data integration platforms. The comprehensive data integration platform refers to all data collection procedures to analyze results and decision-making. Edge gateways are deploying over a large area. The edge computing performed is an overall unbalanced offloading process by the device itself because events and incidence rates are different depending on the area [4,5]. Therefore, edge computing must be balanced through offloading, which distributes tasks and collaborates between edge gateways. Especially in environments such as disasters [6], many unexpected events occur, and some devices are relatively overloaded, so they must quickly prevent overload by interacting with each other.

The overloaded edge gateway chooses the collaboration edge gateway as a closer, more relaxed

Manuscript received January 21, 2021; first revision May 25, 2021; accepted June 27, 2021.
Corresponding Author: Yong-Ik Yoon (yiyoon@sookmyung.ac.kr)
* Dept. of IT Engineering, Sookmyung Women's University, Seoul, Korea (kje9209@sookmyung.ac.kr, xatyna@sookmyung.ac.kr, yiyoon@sookmyung.ac.kr)
** Dept. of Electronics and Information Engineering, Sejong University, Seoul, Korea (hubertkim7@gmail.com)
*** Korea Electronics Technology Institute (KETI), Seongnam, Korea (diesnm201@gmail.com)
Jieun Kang and Svetlana Kim contributed equally to this work.

gateway and offloads the appropriate range. Offloading based on data and compute-intensive applications is the core technology of edge computing [7-10] to extract the appropriate range for offloading. However, the existing offloading system extracts only the possible range without considering the correlation and connectivity between each data or task. The offloading without considering data or task interaction can lead to problems such as data loss, long latency, and security [11] due to repeated data movement.

Edge gateways use common data across multiple spaces and perform computational operations that interact with each other, so the correlation and connectivity of data or tasks are significant. In particular, in models that predict and prevent safety (such as disasters, social security, earthquakes, and construction), the correlation between each data or task is very high. For example, sensor data is time-series data with uncertainty, some of which are region-specific and can re-create through combinations. Also, the output of a task can be the input of other tasks.

There is a massive amount of different data coming in now, and analysis methods are also diversifying. In other words, even for the same data analysis task, it is necessary to consider the unloading range from different angles because the budget and available tools differ depending on the data or analysis system that will be used [12-15]. Therefore, in this paper, we propose a range segmentation offloading method based on edge gateway functions (data, task) to improve response time and save bandwidth in collaborative edge computing. The master edge node monitors all the edge gateways in the proposed way, creates range groups by correlation and communication according to the processing method and tasks, and performs offloading by finding the right collaboration gateway for each optimal segmentation.

This paper organized as follows: Section 2 compares existing data with task-intensive offload methods; Section 3 describes how to extract range groups for the RSDO (range segmentation of dynamic offloading) algorithm; Section 4 proposes the RSDO algorithm that cooperates by extracting the scope to be offloaded and the collaborative edge gateway according to the workflow; Section 5 analyzes the proposed scenario-based algorithm and evaluates it through experiments. Finally, Section 6 concludes this paper.

2. Related Works

In [7], the authors proposes offloading the route search based on the size of the data. The proposed offloading defines a transmission path in which the transmission range is maximized based on the data element size for complete transmission of data elements between nodes in temporary node contact and large data size. Heuristic algorithms are using to find paths with high data transmission probabilities and assign each path the amount of data that maximizes delivery probabilities. However, the data offloading in [7] consider the data size and the contact duration when extracting the transferable range but does not consider the correlation between data and the correlation between tasks.

Li et al. [8] offloads computing-intensive workloads to some computing resources based on independent task scheduling. The authors of [16] show that a single workload can be divide into M independent jobs of the same size, and [8] divides the workload arbitrarily. However, they did not consider the connectivity of tasks in which various workloads combine, not independent tasks. When performing offloading between edge gateways, edge gateways must be assumed to use various data and tasks. In determining the amount of data and fragmenting it, the edge gateways can separate data to use

together. When two or more data are combined to create new data, the edge gateways must perform offloading simultaneously. Also, when the same data is using between tasks, or when the output of a task becomes an input of another task, the edge gateways must perform offloading simultaneously. Therefore, it is necessary to consider what data and tasks are using and whether there is a correlation between them. Table 1 shows the differences between the other papers mentioned and the model considered.

Table 1. Differences from previous studies

Study	Model considered in reference	Models considered in this paper
Lu et al. [7]	The data divide into appropriate amounts by searching for an optimal path based on the size of the data.	When dividing data, consider the association of data.
Li et al. [8]	The workload can be arbitrarily divided.	Considering the relevance of tasks, it can divide into tasks with various subdivisions and dependencies.
Lin et al. [16]	A single workload can divide into *M* equally sized independent tasks.	It assumed that various workloads are combined.

2.1 Architecture of the Intelligent Collaboration System

The overall architecture of "Dynamic Offloading for Collaborative Edge Computing" is described in the previous study [17], shown in Fig. 1. The model has a node that acts as a master node, a second master node, an action node, and an edge node. The master node predicts overload through monitoring and balances the state of the entire edge node through an offloading algorithm. The second node is a replica to prevent the load and loss of the master node. The action node receives the calculated result from all nodes and makes a decision based on the result. Each edge node contains a device and a gateway to which the sensor is connected. The leader role of edge nodes is the convergence of data preprocessing and operations to store, transmit, preprocess and analyze data from sensors to gateways. Based on this architecture, in the next section, the scope of data and task-intensive offloading is divided, and offloading is perform.

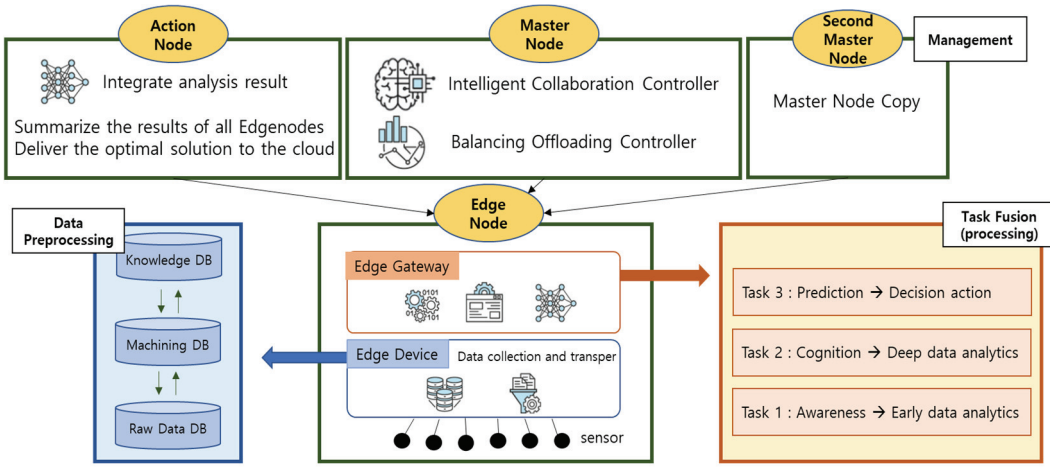


Fig. 1. Edge node collaboration model architecture.

3. Range Segmentation Method by Data and Task-Intensive

Depending on the main function, the active energy of the edge gateway can be defined as follows Eq. (1). Activity energy is the sum of the energy generated by the movement of data and the energy generated by the computation of the data. The memory and CPU usage depending on the amount, size, and analysis techniques of the data. Hence, offloading classifies scopes by considering the correlation of data and tasks according to the gateway function.

$$Activation\ Energy(Memory, CPU) = Data\ Movement\ Energy + Computation\ Energy \quad (1)$$

Correlation and association are challenging to identify without knowing the relationship in advance. It assumed that there are rules or pre-trained rules for each domain.

The master node monitors the number of data variables entering each edge gateway and the size (type) and amount of each variable in real-time. Based on the monitored information, sets are creating by the assumptions set in Sections 3.1 and 3.2.

3.1 Data Correlation Group

First, to know the correlation of the data, the preprocessing needs to find out which data used in which analysis. Correlation with other data can increase as data used in multiple places. Fig. 2 shown Case 1, which offloads a group with many independent data, has low redundancy of the same data on both sides [18].

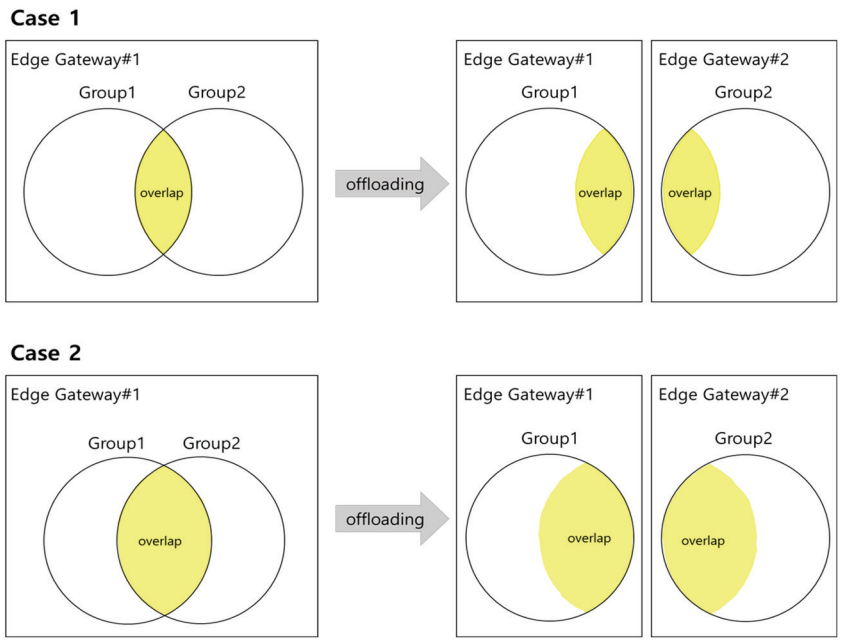


Fig. 2. Two cases for data redundancy after offloading.

In collecting and preprocessing data, the grouping considering the correlation of data created as follows. If the i^{th} data correlated with the data belonging to group k, g_{ki} is assigned a value of 1;

otherwise, this is assigned a value of 0. The G matrix of $k \times i$ represents the memory-intensive group.

$$g_{ki} = \begin{cases} 1 \; if \; data\_i \in group \; k \\ 0 \; otherwise \end{cases}, \; G = \begin{bmatrix} g_{11} & \cdots & g_{1i} \\ g_{21} & \cdots & g_{2i} \\ g_{31} & \cdots & g_{3i} \\ g_{41} & \cdots & g_{4i} \\ \cdots & \cdots & \cdots \\ g_{k1} & \cdots & g_{ki} \end{bmatrix} \tag{2}$$

The G matrix represents all representable data correlation groups. Eq. (3) calculates the space consumed according to the size and amount of data included in each k group included in the G matrix. Data movement energy (DME) represents energy generated when data moves between edge gateways and data, where data preprocessing energy (DPE) is the energy generated during data preprocessing. The energy of each group segmented according to the data correlation depends significantly on the size (type and amount) of the variables used [19]. The calculated value is using to find a suitable collaboration edge gateway after selecting a set that is not larger than the idle space of the collaboration edge gateway.

$$DME_{k,i} + \; DPE_{k,i} = \sum_{i=1}^{n} g_{ki}(Size_{BW_i} * volume_i) + \sum_{i=1}^{n} g_{ki}(Size_{GPU_i} * volume_i) \tag{3}$$

That is, the range extracted using the DME and DPE values shows memory usage and CPU usage. If memory is insufficient depending on the cause of the overload, the offloading range uses the data correlation group.

3.2 Task Connectivity Group

When performing tasks, the following conditions apply when grouping in consideration of the relevance of tasks. Depending on the workflow, t_{ik} is 1 if the output of the i^{th} task is the same as the input of another task, and 0 if not. It can express as a T matrix of $\times \, i$. That can extract the task connection group according to the number of all cases in which t_{ik} has a continuous 1(one) on a row basis. Grouping of task dependencies is the creation of all subgroups that correlated between tasks.

$$t_{ki} = \begin{cases} 1 \; if \; task_{i+1} = next \; of \; task_i \\ 0 \; otherwise \end{cases}, \; T = \begin{bmatrix} t_{11} & \cdots & t_{1i} \\ t_{21} & \cdots & t_{2i} \\ t_{31} & \cdots & t_{3i} \\ t_{41} & \cdots & t_{4i} \\ \cdots & \cdots & \cdots \\ t_{k1} & \cdots & t_{ki} \end{bmatrix} \tag{4}$$

For example, to understand the connectivity structure of the task, we simplify the dependent system into an independent linear sequence processing module, as shown in Fig. 3. A dependency system is a workflow made up of dependent tasks that interact between modules. In a simplified form, three large groups are created, divided into groups of one or more tasks by associating tasks. The algorithm offloads the task connectivity group from the created groups by selecting the group containing the lowest mobility and high priority tasks. The priority depends on the domain, for example, workflow order and order without rushing. Depending on the workflow, the task that is slowest or lacks necessary functionality has the highest priority. Submitting tasks with a low priority creates space for emergency tasks to complete more quickly.

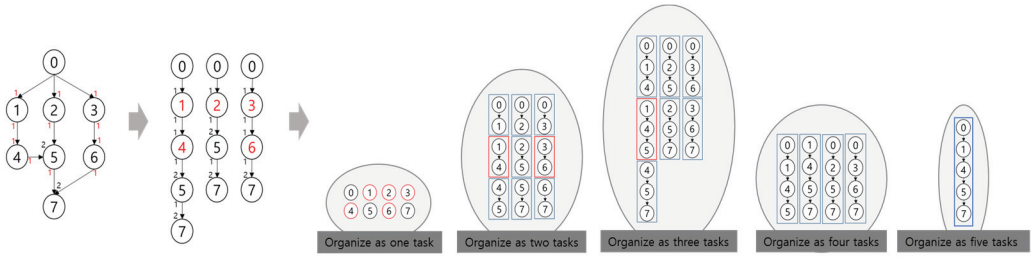


Fig. 3. A simplified example of dependent systems for task connectivity.

Therefore, if a group is created under the following conditions, Eq. (4) calculates the space consumed according to the amount of data contained and the analysis method of the task in each k group of the T matrix. Data movement energy (DME) represents the energy generated by the movement of data between edge gateways. Task computation energy (TCE) represents the energy generated by calculating a task according to an analysis technique. The energy of each group, categorized by task connection, varies greatly depending on the size (type and amount) of variables used and the analytical technique. The Eq.(5) calculation used when searching for a suitable collaborative edge gateway selects a set that is no larger than the idle space of the collaborative edge gateway.

$$DME_{k,i} + TCE_{k,i} = \sum_{i=1}^{n} t_{ki}(Size_{BW_i} * volume_i) + \sum_{i=1}^{n} t_{ki}(Size_{GPU_i} * volume_i) \quad (5)$$

That is, the range extracted using the DME and TCE values shows memory usage and CPU usage. If the CPU is insufficient depending on the cause of the overload, the offloading range uses the task connectivity group. If the time and energy when moving or receiving the result value of the previous task are too large, it is not time efficient to perform it in the overloaded node. Therefore, it is also essential to select a node that will perform and cooperate with the task.

4. RSDO Algorithm

Figs. 4 and 5 show the flow of our proposed RSDO based data and task-linked offloading algorithm. The master node continuously checks the generated data and the operations performed on each edge node through the real-time monitoring process. The monitoring process verifies the type, size, and analysis method of variable data used by the edge gateway to validate each edge node status. It helps identifies the cause of the overload based on CPU and memory utilization. Also, the status information of each edge is necessary to understand the correlation between data or connectivity between tasks.

Memory is used relatively more with a lot of data, and the CPU is used relatively more with more bandwidth. Thus, if high memory is using, it falls back to the workflow shown in Fig. 4. CPU utilization falls back to the offload algorithm, as shown in the workflow in Fig. 5.

The workflow in Fig. 4 extracts the list of offloading ranges using the data correlation grouping method of Section 3.1. The workflow in Fig. 5 extracts the list of offloading ranges using the task relevance grouping method of Section 3.2. The offloading range select according to the edge gateways available for collaboration. The range also must choose the optimal collaboration edge gateway through idle space.

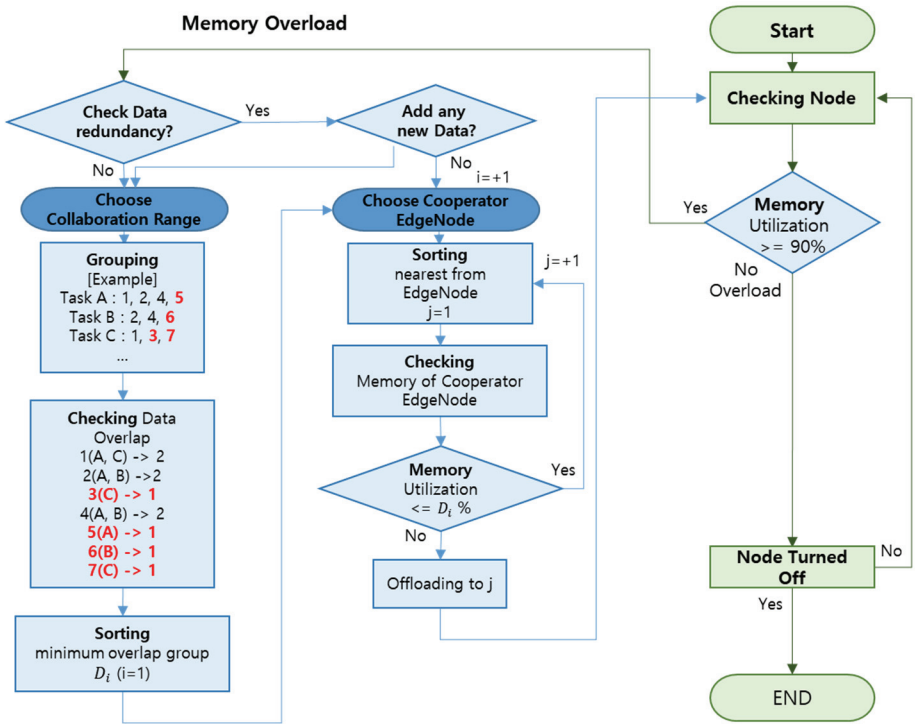


Fig. 4. Flowchart based on data correlation group for collaboration offloading algorithm.

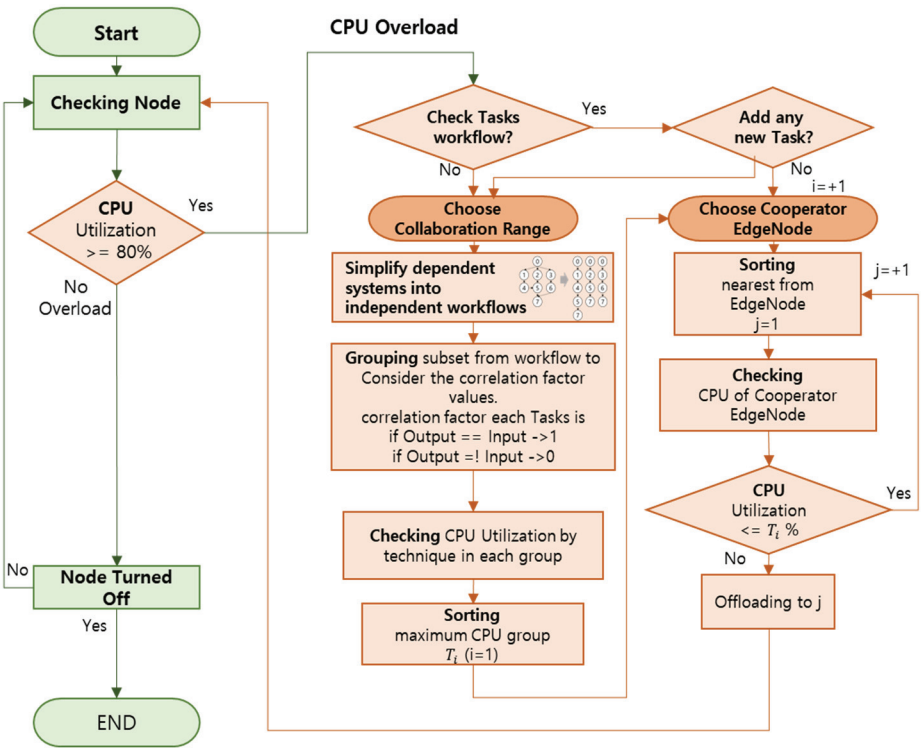


Fig. 5. Flowchart based on task connectivity group for collaboration offloading algorithm.

A few things, such as street, slack, and urgency, consider choosing a collaborative edge gateway. It should not be too far away, but even if they are close, there should be enough space to collaborate. Also, even if there is a lot of space available for collaboration, it should not be a collaboration gateway if it is performing urgent work. With all of this in mind, the master node needs to find a suitable collaborative edge gateway. Algorithm 1 explains how to select an area for cooperation and accordingly find a suitable cooperative node.

4.1 Offloading Range Selection based on Range Segmentation Method Algorithm

The sets grouped for scoping in Section 3 are not all collaboration scopes. The energy used by the collaboration group should be less than the idle space of the neighboring collaboration nodes. In other words, the collaboration group should have less energy than average as much as they can help. Therefore, in Algorithm 1, the input includes the *G* matrix, the *T* matrix, the group energy, and the C_n with output includes the collaborative group. The *G* and *T* matrices are offload groups consisting of g_{ki}, t_{ki}, created by considering data redundancy and mobility. The C_n is a list of gateways in which the idle space has more than a threshold.

Since the group has already considered data redundancy and mobility, priority gives to the least energy. The groups that can be collaborated is to prevent overload by moving from small ones step by step. The ranked group list is essential to use when selecting a collaborative edge gateway in the 4.2 algorithms.

Algorithm 1. Offloading range selection based on range segmentation method

Input: G_i, T_i, $(DME_{k,i} + DPE_{k,i})$, $(DME_{k,i} + TCE_{k,i})$
C_n ← List of gateways that can collaborate with idle spaces above the threshold

Output: Possible Group Data, Possible Group Task
Possible Group Data ← Data-intensive Group with possibility of cooperation
Possible Group Task ← Task-intensive Group with possibility of cooperation

```
For i=1 to k
    If  G_i <= Average of idle space in C_n
        Possible Group Data = G_i
        sorting by "DME+ DPE" ascending

    If  T_i <= Average of idle space in C_n
        Possible Group Task = T_i
        sorting by "DME+ TCE" ascending
End For

G_GroupList(Data_list, Total_Power) = Possible Group Data
T_GroupList(Data_list, Total_Power) = Possible Group Task

Return G_GroupList, T_GroupList
```

4.2 Collaborator Edge Gateway Selection based on Range Segmentation Method Algorithm

Algorithm 2 measures the overload of the current edge gateway to find the cooperative gateway. The group used in relative memory overload is a range of groups classified based on the correlation of the data. Our article defined the threshold as 80% for CPU and 90% for memory. The device can notify that the CPU is at total capacity from the threshold setting or that the device's hard drive is too entire.

Collaboration nodes consider both distance and idle space to find the most suitable gateway. Check idle space sequentially, starting with the nearest collaborative edge gateway. The memory and CPU energy of the possible data group becomes the weight *w*. That is because the collaboration gateway has enough memory, but there shouldn't be a case of running out of CPU. When multiplied by the weights, the group and gateway closest to the average become the collaboration node.

Conversely, when there is a relative overload on the CPU, the group to be used becomes a scope group classified according to task connectivity. Likewise, the memory and CPU energy of the possible data group becomes the weight *w*. The segment groups weighed according to the CPU utilization by the object unit. It is should not run out of memory as well as CPU.

If the edge node is still overloaded after offloading, both algorithms search for the next set and the first edge node of the collaborator. If there is no space left on the first collaborator edge node, the algorithm continues offloading until memory and CPU usage of the border node falls below the threshold. In this case, the algorithm searches the second collaborator edge node. It repeats the collaborative process until the edge node's memory and CPU utilization drops below the threshold. That can improve the performance of the entire function at the edge node and at the same time solve the overhead problem by balancing unbalanced edge calculations.

Algorithm 2. Collaborator edge gateway selection based on range segmentation method

Input: G_GroupList, T_GroupList, $(DME_{k,i} + DPE_{k,i})$, $(DME_{k,i} + TCE_{k,i})$

C_n ← List of gateways that can collaborate with idle spaces above the threshold
G_j ← G_GroupList(Data_list, Total_Power) = Possible Group Data
T_j ← T_GroupList(Data_list, Total_Power) = Possible Group Task,

Output: Collaborator edge gateway

```
For j=1 to n
    If  Current edge gateway overload status = Memory
        w = G_j's memory and CPU usage

    Elif  Current edge gateway overload status = CPU
        w = T_j's memory and CPU usage

   Select gateway with w*C_n (memory, CPU) value close to average energy

End For

Return  Collaborator edge gateway
```

5. Results

The result shows how to create a random variable based on a scenario to set the most appropriate range to reduce redundancy. Tables 2 and 3 show a detailed description of the variables and tasks collected in the experimental environment when five usage data are available. It shows the types of variables collected, the list of variables associated with each variable (data correlation list), the type of task that will perform, and the list of variables used by each task (task connectivity list).

Table 2. Data correlation list

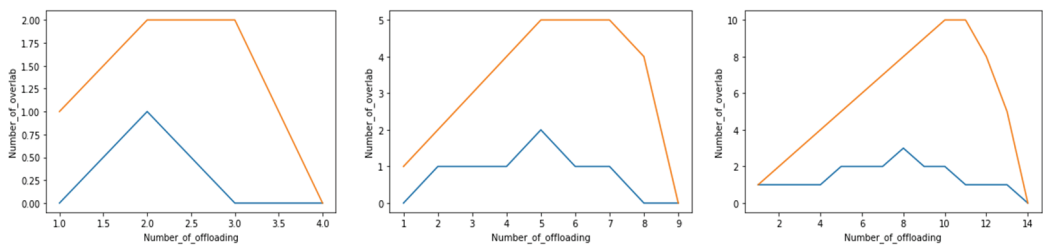
Data type	Data correlation
data1	data2, data4, data5
data2	data1, data4
data3	-
data4	data1, data2
data5	data1

Table 3. Task connectivity list

Task type	Task connectivity
Task1	data3, data4
Task2	data1, data2
Task3	data1, data3 (0utput:data5)
Task4	data5

First, it shows how to optimize the scope of collaboration using collaboration offload that takes all data and tasks into account based on the scenario. As a task-intensive problem, if Task3 wants to offload data1 and data3, the rest of the data is data2, data4, and data5. However, data2, data4, and data5 require data1 for data and task. Hence, when Task3 offloaded, data1 will be duplicated in the edge that collaborates with the existing edge. If data5 is the output data generated by the Task3 calculation, Task4 will use the Task3 result data5, so it will be offloading as if it were judged to be relevant. By sending Task4, data1 keeps duplicates and reduces the number of Tasks running on edge using offload.

Next, we verified that we could reduce the value of data redundancy using three cases: 5 data, 10 data, and 15 data (Fig. 6). The correlation between variables was randomly applied between 0 and 1, and if it was 0.8 or more, the two variables were relevant. In Fig. 6, the x-axis is the number of variables included in the offloading group, and the y-axis is the duplicate value. The duplicate value is the number of



(a) (b) (c)

Fig. 6. Minimum and maximum overlap values according to the unloading range of three cases: (a) 5 data, (b) 10 data, and (c) 15 data.

variables that create all possible groups of offload associations and overlap between unloaded groups and groups that do not. As in the three cases, the number of variables gradually increases. As the tasks become more diverse, overlapping maximum (orange line) and minimum (blue line) data increases. Thereby, selects the group of tasks that must first offload within the scope of the contained collaboration. Then, based on the offload range selection algorithm, selects the group with the fewest duplicate values. The algorithm explains the possibility of sending the same amount of offload range to minimize data redundancy.

6. Conclusion

This paper proposed a real-time monitoring algorithm for identifying the cause of overloaded edge nodes through a hybrid state based on data processing and task performance. The algorithm creates range groups by correlation and connectivity according to the data and task processing method. It identifies any overloading situation in edge computing within a network and performs offloading by searching for an appropriate collaboration gateway for optimal segmentation.

The algorithm examines the data and task dependency problem and extracts optimal offloading range to minimize waste of unnecessary memory space and offloading delay. Future research will be focusing on strengthening the way to construct a collaborative environment by taking into account the features of cooperator nodes. In addition, a new dataset will be prepared to compare the efficiency of this algorithm with the other algorithms

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent technology for adaptive and disposable IoT).

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2019R1I1A1A01064054).

References

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, “A survey on internet of things: architecture, enabling technologies, security and privacy, and applications,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125-1142, 2017.

[2] A. Colakovic and M. Hadzialic, “Internet of Things (IoT): a review of enabling technologies, challenges, and open research issues,” *Computer Networks*, vol. 144, pp. 17-39, 2018.

[3] T. Kawakami, “A structured overlay network scheme based on multiple different time intervals,” *Journal of Information Processing Systems*, vol. 16, no. 6, pp. 1447-1458, 2020.

[4] J. Wang, X. Gu, W. Liu, A. K. Sangaiah, and H. J. Kim, “An empower Hamilton loop based data collection algorithm with mobile agent for WSNs,” *Human-centric Computing and Information Sciences*, vol. 9, article no. 18, 2019. https://doi.org/10.1186/s13673-019-0179-4

[5] L. Zhou and Y. Shan, "Privacy-preserving, energy-saving data aggregation scheme in wireless sensor networks," *Journal of Information Processing Systems*, vol. 16, no. 1, pp. 83-95, 2020.

[6] L. Tran, H. To, L. Fan, and C. Shahabi, "A real-time framework for task assignment in hyperlocal spatial crowdsourcing," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 3, article no. 37, 2018. https://doi.org/10.1145/3078853

[7] Z. Lu, X. Sun, and T. La Porta, "Cooperative data offload in opportunistic networks: from mobile devices to infrastructure," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3382-3395, 2017.

[8] B. Li, M. He, W. Wu, A. K. Sangaiah, and G. Jeon, "Computation offloading algorithm for arbitrarily divisible applications in mobile edge computing environments: an OCR case," *Sustainability*, vol. 10, no. 5, article no. 1611, 2018. https://doi.org/10.3390/su10051611

[9] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," *Software: Practice and Experience*, vol. 44, no. 2, pp. 163-174, 2014.

[10] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: privacy-preserving data aggregation in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, Anchorage, AK, 2007, pp. 2045-2053.

[11] H. He, L. H. Zheng, P. Li, L. Deng, L. Huang, and X. Chen, "An efficient attribute-based hierarchical data access control scheme in cloud computing," *Human-centric Computing and Information Sciences*, vol. 10, article no. 49, 2020. https://doi.org/10.1186/s13673-020-00255-5

[12] H. Kim and S. Lee, "Document summarization model based on general context in RNN," *Journal of Information Processing Systems*, vol. 15, no. 6, pp. 1378-1391, 2019.

[13] M. J. J. Ghrabat, G. Ma, I. Y. Maolood, S. S. Alresheedi, and Z. A. Abduljabbar, "An effective image retrieval based on optimized genetic algorithm utilized a novel SVM-based convolutional neural network classifier," *Human-centric Computing and Information Sciences*, vol. 9, article no. 31, 2019. https://doi.org/10.1186/s13673-019-0191-8

[14] Y. Bengio, *Learning Deep Architectures for AI.* Hanover, MA: Now Publisher, 2009.

[15] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224-2287, 2019.

[16] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," *Software: Practice and Experience*, vol. 44, no. 2, pp. 163-174, 2014.

[17] J. Kang, S. Kim, J. Kim, N. Sung, and Y. Yoon, "Dynamic offloading model for distributed collaboration in edge computing: a use case on forest fires management," *Applied Sciences*, vol. 10, no. 7, article no. 2334, 2020. https://doi.org/10.3390/app10072334

[18] X. A. Yan, W. Q. Shi, and H. Tian, "Cloud storage security deduplication scheme based on dynamic bloom filter," *Journal of Information Processing Systems*, vol. 15, no. 6, pp. 1265-1276, 2019.

[19] C. T. Yang, S. T. Chen, Y. W. Chan, and Y. C. Shen, "On construction of a cloud storage system with heterogeneous software-defined storage technologies," *Human-centric Computing and Information Sciences*, vol. 9, article no. 12, 2019. https://doi.org/10.1186/s13673-019-0173-x

Jieun Kang **https://orcid.org/0000-0003-0679-0317**

She received her BS in Information Statistics from the Hoseo University in 2018. She received her M.S. degree in IT Engineering from the Sookmyung Woman's University in 2021. She is currently an integrated Ph.D. student in the School of IT Engineering at Sookmyung Woman's University, Seoul, Korea. Her current research interests are in the distributed middleware, cloud computing, collaboration edge computing, Bigdata, awareness, cognition, smart service, and fusion sensors.

Svetlana Kim **https://orcid.org/0000-0002-9617-2610**

She received her B.S. and M.S. degrees in multimedia science from the Sookmyung Woman's University in 2005 and 2007, respectively. She received her Ph.D. in Distribution systems from the School of IT Engineering at Sookmyung Woman's University, Korea, in 2017. Since 2019, she has been a research professor at Big Data Using Research Center, Sookmyung Women's University, Seoul, Korea. Her research interests are in the area of ubiquitous computing, distributed middleware, realistic media, cloud computing, edge computing, big data, awareness, cognition, CPS, fusion sensors, and hybrid multi-modal.

Jae-Ho Kim **https://orcid.org/0000-0001-6597-7988**

He is a professor in the department of electric engineering in Sejong University, Seoul, Korea. He served as a director in Autonomous IoT Research Center at the Korea Electronics Technology Institute (KETI), South Korea. He has led research projects for IoT platform, smart city data hub, unmanned aerial vehicle platform, epidemic investigation support system (EISS) for COVID-19 recently. His expertise covers the IoT platforms, intelligent systems and networks. He is now serving as IoT and Smart City Platform Project Group chair of the Telecommunications Technology Association. He received a Ph.D. in the electrical & electronic engineering from the Yonsei University, South Korea. His research interests are in the areas of Internet of Things, smart city and autonomous intelligent systems.

Nak-Myoung Sung **https://orcid.org/0000-0001-9014-7847**

He received his B.Sc. and M.Sc. degrees in electronic and information engineering from the Hankuk University of Foreign Studies, Korea, in 2010 and 2015, respectively. He is currently a researcher at the Autonomous IoT Research Center, Korea Electronics Technology Institute (KETI), South Korea. His research interests include Internet of Things, future internet architecture, standard for M2M/IoT systems and smart city platform.

Yong-Ik Yoon **https://orcid.org/0000-0002-9385-3306**

He received his B.S. in Statistics from the Dongguk University in 1983 and M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1985. From 1985 to 1997, he served as senior researcher at Electronics and Telecommunications Research Institute (ETRI) in following research projects; the research project of the development environment of exchange system, the TDX-10 exchanger development project, the mobile communication development project, and ATM exchange development project. He received his Ph.D. in multimedia science and distribution systems from KAIST in 1994. Since 1998, he has been a professor of Sookmyung Woman's University. His interests include middleware, smart services, IoT, situation awareness, embedded systems, ubiquitous computing, distributed system, real-time processing system, real-time OS/DBMS, big data, fusion multi-sensing, edge computing and digital twin.