


## Article

# Dynamic Offloading Model for Distributed Collaboration in Edge Computing: A Use Case on Forest Fires Management

Jieun Kang <sup>1</sup>, Svetlana Kim <sup>1</sup>, Jaeho Kim <sup>2</sup>, NakMyoung Sung <sup>2</sup> and YongIk Yoon <sup>1,\*</sup> 

<sup>1</sup> Department of IT Engineering, Sookmyung Women's University, 100, Chungpa-ro 47 gill, Yongsan-Gu, Seoul 04310, Korea; kje9209@sookmyung.ac.kr (J.K.); xatyna@sookmyung.ac.kr (S.K.)

<sup>2</sup> KETI (Korea Electronics Technology Institute), 25, Saenari-ro, Bundang-gu, Seongnam-si, Gyeonggi-do 13509, Korea; hubertkim7@gmail.com (J.K.); diesnm201@gmail.com (N.S.)

\* Correspondence: yiyeon@sookmyung.ac.kr; Tel.: +82-2-710-9771

Received: 31 January 2020; Accepted: 25 March 2020; Published: 29 March 2020



**Abstract:** With the development of the Internet of Things (IoT), the amount of data is growing and becoming more diverse. There are several problems when transferring data to the cloud, such as limitations on network bandwidth and latency. That has generated considerable interest in the study of edge computing, which processes and analyzes data near the network terminals where data is causing. The edge computing can extract insight data from a large number of data and provide fast essential services through simple analysis. The edge computing has a real-time advantage, but also has disadvantages, such as limited edge node capacity. The edge node for edge computing causes overload and delays in completing the task. In this paper, we propose an efficient offloading model through collaboration between edge nodes for the prevention of overload and response to potential danger quickly in emergencies. In the proposed offloading model, the functions of edge computing are divided into data-centric and task-centric offloading. The offloading model can reduce the edge node overload based on a centralized, inefficient distribution and trade-off occurring in the edge node. That is the leading cause of edge node overload. So, this paper shows a collaborative offloading model in edge computing that guarantees real-time and prevention overload prevention based on data-centric offloading and task-centric offloading. Also, we present an intelligent offloading model based on several scenarios of forest fire ignition.

**Keywords:** edge computing; offloading computation; distributed collaboration; dynamic offloading; IoT; forest fires

## 1. Introduction

Nowadays, the field of analysis using not only numerical data but also image, video and audio data [1,2] is expanding rapidly due to the development of the IoT and the use of numerous smart/intelligent devices. As data types diversify, the amount of data and operations grow exponentially. In terms of hardware offloading the vast amounts of data and computing tasks into the cloud is efficient to work but it creates problems such as network delays, increased energy consumption and so on [3–7].

Much research has been done to address these issues [8–12]. Matters to consider within many papers are response time requirement, battery life constraint, bandwidth cost-saving as well as data safety and privacy [13,14]. Recent researches have focused on saving energy and minimizing computation through data processing, traffic management, job placement and resource allocation. Ref. [15,16] leverage the edge cloud to overcome the limitations of the limited computing capacity of mobile devices and to avoid high latency due to offloading operations. Ref. [17] uses fog computing to minimize data processing time and latency through data-intensive computing offload. Ref. [18] uses

task-intensive offload to limit battery life to minimize average work time. Ref. [19] is also a study of edge computing with a focus on computational offloading. The edge computing performs some or all of the data and tasks at the edge node [20–22] without transferring all data generated at the end node to the cloud to optimize energy consumption. Processing bigdata and tasks at the edge node without going to the cloud can avoid long delays. As described above, many types of research on edge computing have proposed efficient strategies through offloading between devices, edge nodes, and the cloud.

With the recent growth of intelligent and mobile devices combined with technologies such as IoT and Augmented Reality(AR), the focus has shifted to getting real-time responses with situational awareness. Edge computing makes processing faster than cloud computing when the field of real-time responses is increasing and requires immediate analysis [23]. However, the biggest problem with edge computing is the overload of edge nodes. Failure problems such as discharges of electricity can result in significant losses, which are more severe than the issue of delay.

In this paper, the offloading based on edge computing divide main functions into tasks that require data storage and computation. The method of related works described above only considers data or task offloading to save energy and reduce latency. As data grows, the compute volume increases at the edge node, and tasks of offloaded to the cloud due to limited computing capacity. This solves the computational complexity but once again fails to respond in real-time due to latency and bandwidth issues. If you only consider one of the data and tasks, this problem will continue to repeat. Therefore, two functions result in congestion (overload problems due to centralized offload) as well as network and speed delays. These problems are especially important in areas where an immediate response is needed to detect nearby objects, collect real-time data, and speed up complex calculations [24,25]. To identify abnormal situations and to take quick warnings and actions, a hybrid approach is needed that analyzes multiple volumes of sensor data in real-time [8]. However, if the edge node is overloaded, the performance of the function is delayed, which makes it difficult to guarantee safety. Existing methods are very inefficient for a fast way to deal with emergencies. For example, if a fire detected in a mountain or forest, it must be quickly suppressed to prevent forest fires, and in the event of a fire, the propagation path must track to avoid further loss.

In recent years, the seriousness of wildfires is rapidly increasing. Currently, much of the damages caused by forest fires are extensive in many countries including Korea [26]. Last year, a fire on the roadside spread to the mountains, causing a huge fire in South Korea. The high temperature, as well as dry and strong wind speed according to the local characteristics were the cause of the large forest fire. The changing wind direction was difficult to predict the diffusion range and speed. Recently, the prolonged forest fires in Australia are also causing enormous damage, with about 50,000 square kilometers turning to ashes and the loss of more than 500 million animals and plants. The changing climate has increased the frequency of droughts resulting in massive wildfires. A lack of responsiveness to climate change does not prevent the spread of forest fires, and the weak firefighting capacities have no choice but to see the situation of the damage increase. The most important things are to make predictions and respond quickly before an event occurs. This can reduce the amount of damage even if there are not enough firefighters or poor firefighting abilities. Events that occur in nature are difficult to predict, so forest fires require a variety of data usage and in-depth analysis. That is means that the data becomes infinite and the analysis technique becomes more complicated for prediction. Environmentally, wildfire creates a situation where the edge node cannot prevent overload, and this is why the proposed offloading model should be used.

In this article, two subtasks used to solve the problem of congestion: collaboration on offloading data and collaboration on offloading tasks to optimize the problem. There is no comprehensive study covering both data and task aspects. As far as we know, this is the first study in which an attempt has made to prevent overload, taking into account the focus on both data and tasks.

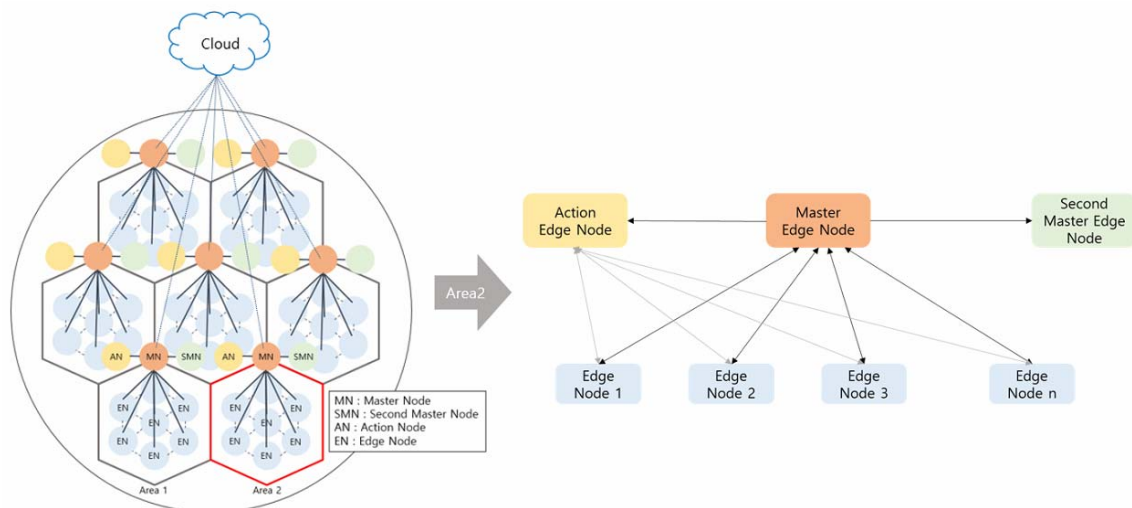
The rest of the paper organized as follows: Section 2 describes the architecture of Intelligent collaboration for computation offloading. Section 3 contains algorithms based on the proposed

intelligent offloading model with the scenario of the forest fire. Section 4 describes the feature and definition problem of edge computing. The conclusions of this paper include in Section 5.

## 2. Architecture of Intelligent Collaboration for Computation Offloading

This section describes the architecture of the computation offloading model in edge computing. The edge nodes collaborate with other nodes to solve the overload problem that occurs due to inefficient partitioning and trade-off. We propose a model that considers both data and tasks for effective offloading.

Figure 1 shows the architecture of the intelligent collaboration system proposed in this paper. The whole is divided equally among  $n$  areas. Each area has  $n$  edge nodes, one master node, one is the second node, and another one is the action node. The edge node is requesting collaboration to the master node by predicting the overload that occurs in the process of collecting, storing, analyzing, and determining data and reporting the results. The master node manages efficient collaboration by monitoring the condition of all edge nodes in real-time and providing edge nodes to collaborate with an overloaded edge node. The second node acts as an assistant of a master node in the case of overload and failure. Lastly, it is an action node that receives the prediction result and responds quickly. The specific roles for each node for distributed collaboration scheduling algorithm are as follows.



**Figure 1.** Architecture of the intelligent collaboration system.

### 2.1. Edge Node

The edge node classifies unnecessary data through the process of collecting and preprocessing data generated from the end node and creates new data based on knowledge of the domain. Next, report the analysis result to the cloud after analyzing and processing. On the whole, as the amount of data and the number of tasks increases, the overload possibility at the edge node increases. Therefore, the edge node predicts the occurrence of overload revolved around data and tasks and requests collaboration at the master node. The edge node transmits some of the data and tasks that are classified based on data and task-centric algorithms to a collaborative edge node when getting the IP address of another edge node that can collaborate from the master node.

### 2.2. Master Node

Every edge node can be a master node. In consideration of the status of the edge node in real-time, the edge node with the least likely to overload becomes the master node. The master node is responsible for distributed collaboration schedule management through monitoring all edge nodes in real-time for collaborative management between edge nodes [27]. First, to analyze the distance between the

edge node that made a collaboration request and  $n-1$  edge nodes. The farther distance between edge nodes when they communicate to transfer data, the latency and energy consumption are increased. The edge node with the closest distance should be given relatively large weight because it can quickly derive from data transfer to results. That is because the cumulative amount of data and tasks increases, increasing the possibility of overload. Although cost and performance alone [28] can be offloaded at the lowest cost and high performance, it is important to analyze the potential for overload to consider how long collaboration can be maintained [29].

Therefore, the edge node can maintain the most extended collaboration, and the lowest latency selected as a cooperative node. The IP address of the edge node chosen delivered to the overloaded edge node, which immediately sends data and tasks to resolve the overload. As described above, the master node manages cooperative offloading schedules between them to prevent overloading of all edge nodes in each region. In detail, we will describe the collaborative offloading algorithm divided into data and task-centric, which is the basis of the collaboration model.

### 2.3. Second Master Node

The second master node plays a supporting role for the master node. The possibility of sudden failure or overload of the master node is considered. All information of the master node is transmitted to the second master node in real-time. If there is a problem with the master node, the second master node takes over the role of collaborative scheduling management.

### 2.4. Action Node

The action node collects the result analyzed by the edge node. The intelligent system of the action node performs situational response and quick control based on all analysis results.

## 3. Intelligent Dynamic Offloading Algorithm

We defined intelligent collaboration models as four types of models in Section 2. Section 3 defines the algorithm applied to the model. The model collaborates intelligently based on data and task centric algorithms. We define the point of overload and the collaboration algorithms on two bases.

### 3.1. Data Centric Offloading Algorithm

The functions that are offloaded based on data are as follows:

1. Raw data collection from disposable IoT device.
2. Knowledge information collection from awareness sensing and platform.
3. Awareness fusion base on procession.

The point of overload at the center of data is as follows:

$$\begin{aligned} & \text{Gateway load} \\ & = \text{if } \sum_{i=1}^n (\text{SensingData}_i + \text{Knowledge}_i + \text{DataPreprocessing}_i) = \text{Threshold}, \end{aligned} \quad (1)$$

While all three are being done at the same time, if the amount of energy they consume exceeds the threshold, an overload is detected (Equation (1)).

First of all, the data-centric algorithm must consider the relationships between data when overloading occurs. Generally, the more has relevant data, the higher the amount of data that needs to send. That is because all relevant data must be delivered with the cooperation request.

Next, consider the number of data. The higher the amount of data, the more time and energy needs for sending data [30,31]. Therefore, the group with the minimum number of relevant data and the total number of data is first sent to the cooperating node. If the energy consumption quantity is still above the threshold, the next lowest amount of data is sent to the cooperating node. Iterates until it is less than the threshold and stops requesting cooperation when it is less than.

### 3.2. Task Centric Offloading Algorithm

The functions that are offloaded based on task are as follows:

1. Analyze.
2. Cognition.
3. Decide.

The point of overload at the center of task is as follows:

$$Gateway = \text{if } \sum_{i=1}^n (Analyze_i + Cognition_i + Decide_i) = Threshold, \quad (2)$$

While all three are being done at the same time, if the amount of energy they consume exceeds the threshold, an overload is detected (Equation (2)).

First of all, the task-centric algorithm requests collaboration based on the importance of the task when an overload occurs. Case of significant responsibilities(task), maintaining the analysis on own edge node will yield the fastest results. Passing the least important tasks to the collaboration node frees up space for analyzing significant tasks. The criteria for selecting the most vital task will vary depending on the domain.

The next thing to consider is the relationship and the amount of data per task. If a task contains several kinds of data, even if one of the task requests collaboration, the amount of data becomes too large. The more relevant data, the higher the task speed, energy consumption, and the complexity of calculating the task. Therefore, the task with the least data relationship is first moved to the collaborative edge node. Like data-centric, iterates until it is less than the threshold and stops requesting cooperation when it is less than.

### 3.3. Forest Fire Prediction

In this context, this paper classifies works from data collection to analysis for forest fire prediction and spread prevention based on two proposed offloading.

The works at the data centric are as follow:

- Real-time local data collection through forest fire sensor.
- Create a forest fire knowledge database.
- Data preprocessing.

Data collected in real-time by the sensor mainly include temperature, relative humidity, carbon dioxide, and wind direction. Also, it collects historical data such as seasons and weather and knowledge data such as mountainous terrain from external platforms. Finally, the database is designed at each boundary node by preprocessing the data, such as removing duplicate data and creating new data by merging the data.

The works at the task centric are as follow:

- Fire prediction.
- Wildfire spread area (direction) and velocity prediction.
- Forest fire type prediction.
- Prediction of embers landing points.
- Prediction of fire after landing.

The above tasks are performed in the same order as above and analyzed using real-time local data, collected knowledge data, and combined new data. Since the tasks progress sequentially, fire prediction is the most important, and the bottom is the least important.

Unfortunately, the edge node, which performs all of these tasks, is overloaded that occurs due to rapid weather changes, a large amount of data and simultaneous execution of the tasks of forest fire

prediction and spread prediction. Thus, a relaxed edge node collaborates with other nodes to prevent damage or reduce the scope through quick analysis and response.

### 3.4. Scenario of Computation Offloading for Wild Fire

This section shows an example of a task-centric algorithm using virtual data related to the forest fire. Depending on the performance of the edge node, latency, and energy costs may vary. The example demonstrates task collaboration by further subdividing the five tasks that are broadly divided above. This paper deals with nine types, and the number of tasks varies depending on what you are expecting.

Table 1 shows the nine tasks and the data used in each. The tasks are listed in the order of analysis and it is assumed to be a uniform the number of data in each task. Edge nodes commonly perform the tasks in Table 1.

**Table 1.** Tasks for fire and spread range prediction, Part of the data used.

Task	Data
Task 1. Fire prediction	temperature, humidity, fuel, mountain terrain, etc.
Task 2. Calculate spread rate	fuel, mountain terrain, weather, forest physiognomy etc.
Task 3. Forest fire intensity calculation	calculated diffusion rate
Task 4. Calculate flame height	calculated forest fire intensity
Task 5. Types of fire forest	temperature, humidity, forest physiognomy, wind speed etc.
Task 6. Calculate flame distance	wind speed, calculated diffusion rate
Task 7. Prediction of spread area	temperature, humidity, mountain terrain, forest physiognomy, wind speed etc.
Task 8. Calculate the location of the flame landing	wind speed, wind direction etc.
Task 9. Fire predictions at landing locations	temperature, humidity, fuel, mountain terrain, etc.

Figure 2 shows re-arranged in order based on an algorithm to find out data relations between tasks. As mentioned above, Task 1 is the most fundamental analysis, so it should be performed on its edge node. Therefore, the more the data is related to task 1, the higher the importance.

9. Fire								
1. Fire Prediction	Predictions at Landing Locations	2. Calculate Spread Rate	7. Prediction of Spread Area	5. Types of Fire Forest	8. Calculate the location of the flame landing	6. Calculate flame distance	3. Forest Fire Intensity Calculation	4. Calculate Flame Height
Temperature	Temperature		Temperature	Temperature				
Humidity	Humidity		Humidity	Humidity				
Fuel	Fuel	Fuel						
Mountain terrain	Mountain terrain	Mountain terrain	Mountain terrain					
Weather	Weather	Weather						
Forest	Forest	Forest	Forest	Forest				
Physiognomy	Physiognomy	Physiognomy	Physiognomy	Physiognomy				
			Wind speed	Wind speed	Wind speed	Wind speed		
			Wind direction		Wind direction			
			Calculated Diffusion Rate		Calculated Diffusion Rate	Calculated Diffusion Rate	Calculated Diffusion Rate	
				Calculated Forest Fire Intensity				Calculated Forest Fire Intensity
			Calculated flame distance		Calculated flame distance			

**Figure 2.** Tasks re-arranged in order based on algorithm.

There are a total of four tasks related to ‘Task 1. Fire prediction’:

- Task 9. Fire predictions at landing locations
- Task 2. Calculate spread rate
- Task 7. Prediction of spread area



- Task 5. Types of fire forest

The next criterion to look at is the relationship of between data; the more relationships between data, the lower the importance. Also, since the same data is assumed to have the same number of data, the smaller the data used, the lower the influence.

The remaining tasks, listed in order of little relationship between data, are as follows:

- Task 4. Calculate flame height
- Task 3. Forest fire intensity calculation
- Task 6. Calculate flame distance
- Task 8. Calculate the location of the flame
- Task 9. Fire predictions at landing locations

Therefore, when an edge node is overloaded, it is delivered to the edge node, which cooperates in the order of Task 4, Task 3, Task 6, Task 8, Task 5, Task 7, Task 2 and Task 9.

The difference between the proposed algorithm and the existing offloading method in this paper is as follows. The current offloading method offloads some tasks that consume less energy without considering the relationship between data and task. They simply looked at the energy efficiency of the device and offloaded it. However, this method is problematic when applied to forest fires. The importance of task changes depending on the situation and energy consumption also depends on the amount of data. If it merely thinks about energy efficiency and offloads the task, the execution of the important task can be delayed. For example, Task 1 should be monitored at all times to prevent forest fires. However, if a forest fire has already occurred, tasks that prevent spread are more important than Task 1. Task1 covers the most data and is fundamental analysis, so it is good always to do task1 under the edge node's self. However, the importance can vary depending on the situation.

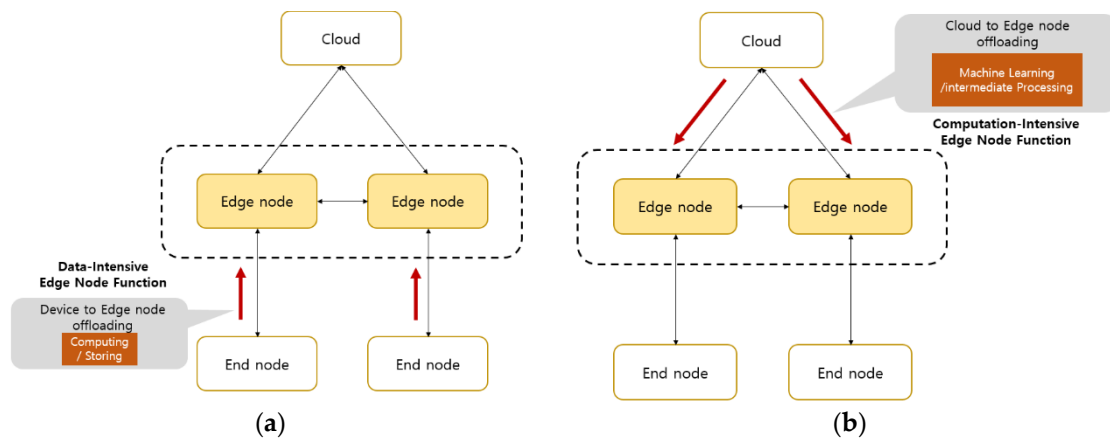
#### 4. Implement and Discussion of Intelligent Dynamic Offloading Model

This section discusses why we proposed an intelligent collaboration model and why to use the model for forest fires. First, Section 4.1 describes offloading based on edge computing according to two functions. Section 4.2 explains in detail the cause of the overload problem due to the two offloading task. The intelligent collaboration model proposed in this paper solves these problems. Finally, Section 4.3 shows the overview processing of the model using some data from Section 3.4.

##### 4.1. Offloading Based on Edge Computing

As described above, the functions offloaded to the edge node are divided into two core technologies: (1) data-intensive offloading and (2) computation-intensive offloading [32]. Figure 1 shows that it has two functions (1) Data and (2) Task.

Data-intensive edge node: As shown in Figure 3a, the data is collected in real-time from the end node stored in the edge node without sending it to the cloud after data pre-processing [33], i.e., only the necessary data submitted to the cloud. Cloud computing compensates for network bandwidth limitations and minimizes latency. [34] With data offloading that pre-processes and filters the sensed data function, only sensitive data is stored, transmitted and sent to the edge node. Data offloading pre-processes and filters the data collected at the edge node close to the location of the device where the data generated. That has the advantage of optimizing cloud computing systems [35] by reducing network traffic (burden) that occurs when sending data sensed from edge devices to the cloud.

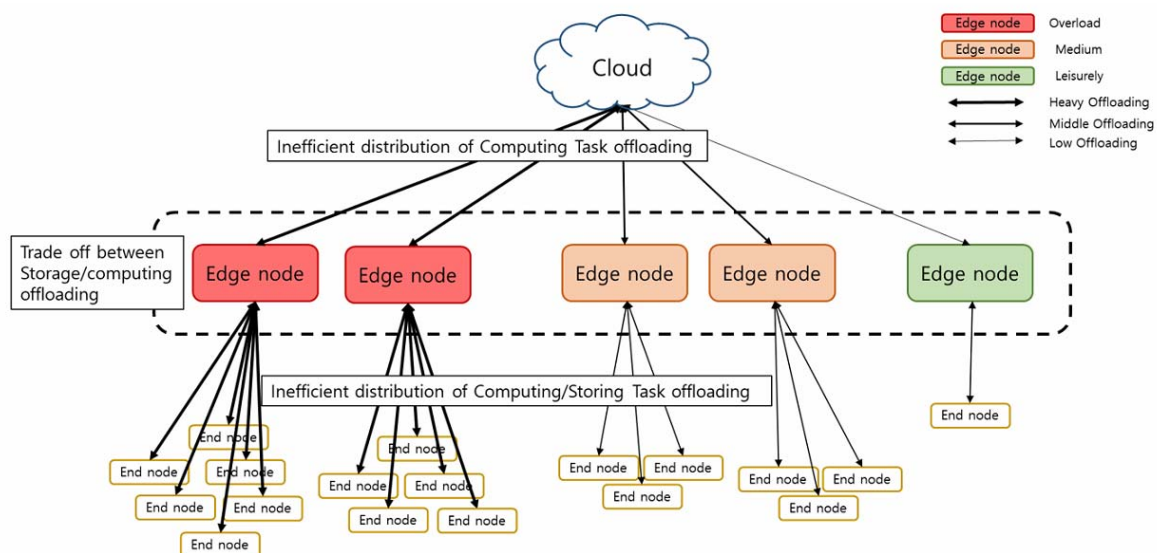


**Figure 3.** (a) Data-intensive edge node function (b) Computation-intensive edge node function.

Task-intensive edge node: Figure 3b shows that offloading computing works for high-performance tasks such as analyzing and processing data from the cloud to the edge node. It leads to an improvement in the quality of calculations, including the delay and power consumption of the device [36]. The advantage is that the real-time response to the situation reaction is more guaranteed [34] by avoiding the significant delay caused by minimizing the network delay time. To date, energy consumption has increased rapidly, like two in the end node and the cloud is unloaded at the border node and centralized. Also, there is a high probability of overloading equipment located on edge. There is a problem in that the processing speed of the calculation task reduced. Therefore, the problem that occurs when two tasks occur at the same time should be considered.

#### 4.2. Problem of Offloading Based on Edge Computing

Both offloading are centralized to the edge node at the same time. Figure 4 shows two significant problems when tasks distributed without considering data offloading and task offloading at the same time—Inefficient partitioning results in a tradeoff.



**Figure 4.** Problems of offloading based on the edge computing.

Inefficient distribution computing/storing offloading: Edge computing means offloading work to the edge node closest to the terminal from which data is generated. Thus, if an event occurs at a location, data will surge at all end nodes around that location and this will be concentrated the data and computing tasks are collected to the nearest edge node. When faced with this situation, offloading of



storing/computing tasks is focused only on specific nodes as shown in the figure. Therefore, inefficient splitting of storing/computing offloading occurs when an overload occurs only at certain edge nodes and some nodes are relaxed. As a result, overload occurs only at certain edge nodes, and some nodes save too much space, resulting in inefficient partitioning of storing/computing offload.

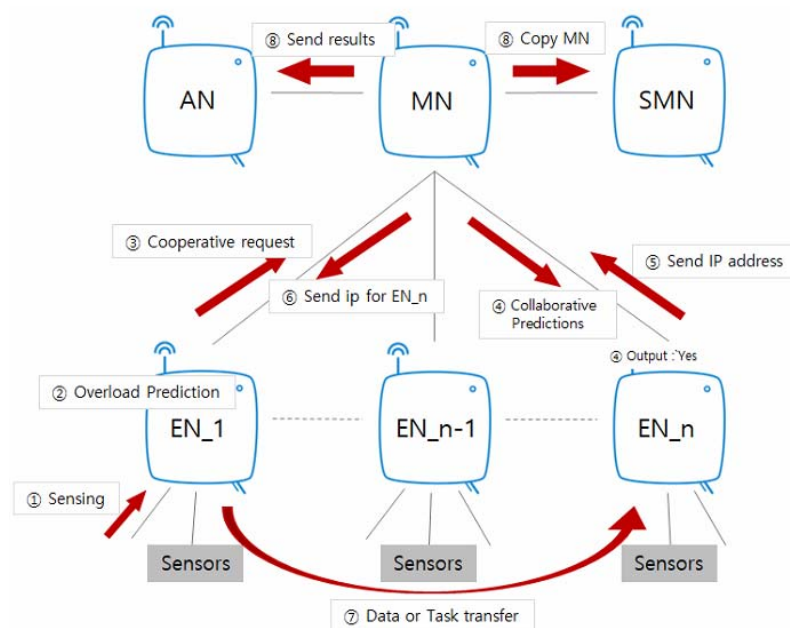
Trade-off between storage/computing offloading: Inefficient partitioning problems also lead to issues with a trade-off. If the data storage functions offloaded to a particular node are large, there is not enough space for computing functions. On the contrary, when more computing functions become available, there is not enough space for data storage functions. This causes delays due to the inability to do some work and cannot guarantee the real-time advantage of edge computing.

Therefore, this paper proposed an intelligent collaboration model with a focus on solving two problems. The model solves problems and distributes results in real-time by dispersing functions through the proposed collaboration algorithm.

#### 4.3. Proposed Model Processing Overview

Section 4.3 shows the flow of collaboration when wildfires occur in certain areas and overloads occur. Based on the proposed model, this is a flowchart to collaborate when an edge node is overloaded in a specific area.

Figure 5 shows the process of the intelligent offloading model for forest fire prediction and prevention scenarios.



**Figure 5.** Dynamic offloading model processing overview.

The cooperative process in case of overload is as follows:

1. Sensing data from sensors of EN\_1.
2. If overload prediction of edge node#1, the result of analysis by the master node.
  - Find the problem by separating it into data and task.
3. Cooperative request to master node.
  - If problem of data intensive offloading, then use the 'Data centric algorithm'
  - If problem of task intensive offloading, then use the 'Tata centric algorithm'
4. MN checks the possibilities of collaborative predictions by data and tasks from edge node#n.

5. If the output is Yes, master node check the IP address of edge node#n.
6. Send IP of EN\_n to edge node#1.
7. Data or task transfer to EN\_n.
  - If problem of data offloading, then use the 'Data centric algorithm'
  - If problem of task offloading, then use the 'Tata centric algorithm'
8. Finally, send all results to action node and copy master node to second master node.

We describe the process by applying scenarios to the cooperative processes listed from 1 to 8. The scenario is when the temperature rises rapidly, and the wind direction changes frequently. First, EN\_1 collects and processes data in real-time. EN\_1 collects and processes data in real-time through sensors. Task1 starts to predict the forest fire when an event that rapidly increases in temperature occurs. The assumption is that the initial fire suppression failed. The following tasks performed to predict the diffusion range and speed through data such as wind direction and wind speed. EN\_1 rapidly increases the workload. The master node monitors all edge nodes in real-time and predicts the overloaded if the energy of EN\_1 increases and exceeds the threshold. EN\_1 is the overload, and the EN\_1 requests cooperation from the Master Node to find the cause of the overload first. The master node divides into data and tasks to find edge node problems. The two algorithms decide the part to collaborate based on the causes of data collection and processing and task analysis. For example, in the case of a task offload problem, use the 'Task-centric algorithm'. The algorithm rearranges the parts to collaborate as in the example in Section 3.4.

The fire prediction task is the most frequently performed task as sensing data is collected. Most fundamentally, to extinguish forest fire through fire prediction will prevent the problem of spreading. The next most important task is to predict the flow of forest fires by using the wind direction, wind speed and temperature in real-time. Therefore, the two tasks are used most frequently and handle the most data, so make sure to analyze them at their edge nodes so that there is no delay. Also, the task of predicting the type of wildfire is the most challenging situation to predict the spread of wildfire. The embers blow in the wind and create a new forest fire. The fireball attached to the upper part of the pine heats the surrounding air, rises above the air. If that meets the strong winds, these embers can fly for 1–2 km. There is the possibility of a considerable risk of fire spreading in a short time and becoming hard to control. To extinguish, it is urgent to analyze the landing area and how quickly it spreads.

Therefore, tasks are important in order of forest fire prediction, diffusion prediction, and forest fire type prediction. The importance may vary depending on the situation. Currently, the temperature is rising and the wind speed is rising, so there is already a forest fire. In this situation, the spread prediction and forest fire type prediction becomes more important than the forest fire prediction task in EN1. After that, consider the relationship and amount of data as in Section 3.4. At the same time, the Master Node analyzes other edge nodes to check the possibility of collaboration by data and work. It checks the possibilities from the edge node where collaboration is most efficient. If it outputs that it can collaborate at edge node #n, the master node checks the IP address and sends the IP to edge node 1. Edge node#1 sends data and tasks to collaborate based on an algorithm to edge node n via IP. If the overload lasts, other data and tasks are sent to the second collaborative edge node. If the overload is resolved, the collaboration request will stop.

This works the same way for other edge nodes. If there is a high possibility that forest fires are likely to spread to EN\_3 according to the wind direction data of EN\_1, the EN\_3 is excluded from the collaborative edge node and analyzes the forest fire prediction from its data. When overloaded, the EN\_3 requests collaboration in the same way.

## 5. Conclusions

The continued increase in data has a more significant impact on the latency and energy consumption of edge computing. The development of big data analytics and intelligent devices increases the amount

of computation for edge nodes, which is related to the latency and energy consumption of edge computing. This centralization of data storage and computing power results in overloading and lowers efficiency.

In this context, we proposed a new intelligent offloading model that helps each other between edge nodes focusing on offloading technologies centered on data and tasks. To our knowledge, this is the first study of intelligent task distribution offloading, which focuses on data and tasks and cooperates between edge nodes. This model solves the overload caused by the limited capacity of the edge node while maintaining real-time benefits.

Therefore, this paper describes a model based on wildfire scenarios that require rapid response and analysis. The process of applying the model and algorithm was suggesting the overload occurring in the process of predicting fire and predicting the spread of forest fire is presented in detail. Therefore, this scenario can prevent damage or reduce the scope of damage by quick analysis and response by collaborating with a relaxed edge node through the proposed model. In other real-time response areas such as disasters and emergencies, the proposed model can be useful for various studies.

In this paper, virtual data is used instead of real data. In future research, we will investigate the function of hardware to specify energy consumption, execution time, etc. and further organize it using virtual data based on forest fire scenarios. Through this, we want to compare the efficiency by analyzing the waiting time and energy consumption when we randomly collaborate and apply the proposed model.

**Author Contributions:** Conceptualization, J.K. (Jieun Kang), S.K., J.K. (Jaeho Kim), N.S. and Y.Y.; methodology, J.K. (Jieun Kang), S.K. and Y.Y.; resources, J.K. (Jieun Kang); writing—original draft preparation, J.K. (Jieun Kang); writing—review and editing, S.K.; supervision, Y.Y.; project administration, J.K. (Jaeho Kim), N.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** Korea government(MSIT) and National Research Foundation of Korea (NRF).

**Acknowledgments:** 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 Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2019R111A1A01064054).

**Conflicts of Interest:** Authors declare no conflicts of interest.

## References

- Jo, B.; Suh, D.Y. Mobile Energy Efficiency Study using Cloud Computing in LTE. *J. Broadcast Eng.* **2014**, *19*, 24–30. [\[CrossRef\]](#)
- Evans, D. The Internet of Things: How the next evolution of the Internet is changing everything. *J. Sens. Actuat. Netw.* **2011**, *1*, 1–11.
- Habib ur Rehman, M.; Jayaraman, P.P.; Malik, S.U.R.; Khan, A.U.R.; Medhat Gaber, M. Rededge: A novel architecture for big data processing in mobile edge computing environments. *J. Sens. Actuat. Netw.* **2017**, *6*, 17. [\[CrossRef\]](#)
- Akhbar, F.; Chang, V.; Yao, Y.; Muñoz, V.M. Outlook on moving of computing services towards the data sources. *Int. J. Inf. Manag.* **2016**, *36*, 645–652. [\[CrossRef\]](#)
- Li, C.S.; Darema, F.; Chang, V. Distributed behavior model orchestration in cognitive internet of things solution. *Enterp. Inf. Syst.* **2018**, *12*, 414–434. [\[CrossRef\]](#)
- Jo, J.H.; Sharma, P.K.; Sicato, S.; Costa, J.; Park, J.H. Emerging Technologies for Sustainable Smart City Network Security: Issues, Challenges, and Countermeasures. *J. Inf. Process. Syst.* **2019**, *15*, 765–784.
- Hossain, M.D.; Sultana, T.; Layek, M.A.; Hossain, M.A.; Hossain, M.I.; Woo, S.Y.; Huh, E.N. Dynamic Task Offloading for Ultra-Dense Mobile Edge Computing. *J. KIISE* **2019**, *2019*, 252–254.
- Singh, S. Optimize cloud computations using edge computing. In Proceedings of the International Conference on Big Data, IoT and Data Science (BID), Pune, India, 20–22 December 2017; pp. 49–53.

9. Thangam, M.S.; Vijayalakshmi, M. Data-intensive Computation Offloading using Fog and Cloud Computing for Mobile Devices Applications. In Proceedings of the International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 13–14 December 2018; pp. 547–550.
10. Masip-Bruin, X.; Marn-Tordera, E.; Tashakor, G.; Jukan, A.; Ren, G.J. Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems. *IEEE Wirel. Commun.* **2016**, *23*, 120–128. [\[CrossRef\]](#)
11. Vallati, C.; Virdis, A.; Mingozzi, E.; Stea, G. Mobile-edge computing come home connecting things in future smart homes using LTE device-to-device communications. *IEEE Consum. Electron. Mag.* **2016**, *5*, 77–83. [\[CrossRef\]](#)
12. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [\[CrossRef\]](#)
13. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [\[CrossRef\]](#)
14. Hong, M.; Jinman, J.; Junyoung, H. A Function Level Static Offloading Scheme for Saving Energy of Mobile Devices in Mobile Cloud Computing. *J. KIISE* **2015**, *42*, 707–712.
15. Tong, L.; Li, Y.; Gao, W. A hierarchical edge cloud architecture for mobile computing. In Proceedings of the 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
16. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [\[CrossRef\]](#)
17. Dolui, K.; Datta, S.K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
18. Chen, M.; Hao, Y. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [\[CrossRef\]](#)
19. Zhang, S.; Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656.
20. Park, M.G.; Zhe, P.Z.; La, H.J.; Kim, S.D. Practical Offloading Methods and Cost Models for Mobile Cloud Computing. *J. Internet Comput. Serv.* **2013**, *14*, 73–85. [\[CrossRef\]](#)
21. Piao, Z.Z.; Kim, S.D. A Prediction-based Dynamic Component Offloading Framework for Mobile Cloud Computing. *J. KIISE* **2018**, *45*, 141–149. [\[CrossRef\]](#)
22. Cho, Y.; Cho, D.; Paek, Y. SorMob: Computation Offloading Framework based on AOP. *KIPS Trans. Comput. Commun. Syst.* **2013**, *2*, 203–208. [\[CrossRef\]](#)
23. Sharma, S.K.; Wang, X. Live data analytics with collaborative edge and cloud processing in wireless IoT networks. *IEEE Access* **2017**, *5*, 4621–4635. [\[CrossRef\]](#)
24. Danial, S.N.; Smith, J.; Khan, F.; Veitch, B. Situation awareness modeling for emergency management on offshore platforms. *Hum.-Cent. Comput. Inf. Sci.* **2019**, *9*, 37. [\[CrossRef\]](#)
25. Aazam, M.; Huh, E.-N. E-Hamc: Leveraging fog computing for emergency alert service. In Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, USA, 23–27 March 2015; pp. 518–523.
26. Mahmoud, M.A.I.; Ren, H. Forest Fire Detection and Identification Using Image Processing and SVM. *J. Inf. Process. Syst.* **2019**, *15*, 159–168.
27. Min, H.; Kim, B.; Heo, J.; Jinman, J. An Offloading Scheme for Reliable Data Processing of Swarm-drones. *J. KIISE* **2018**, *45*, 990–995. [\[CrossRef\]](#)
28. Gai, K.; Qiu, M.; Zhao, H.; Liu, M. Energy-aware optimal task assignment for mobile heterogeneous embedded systems in cloud computing. In Proceedings of the 2016 IEEE 3rd international conference on cyber security and cloud computing (CSCloud), Beijing, China, 25–27 June 2016; pp. 198–203.
29. Kumar, K.; Liu, J.; Lu, Y.H.; Bhargava, B. A Survey of Computation Offloading for Mobile Systems. *Mob. Netw. Appl.* **2013**, *18*, 129–140. [\[CrossRef\]](#)
30. Dang, T.N.; Hong, C.S. A Non-convex ADMM Approach for Data Offloading in Mobile Edge Computing. *J. KIISE* **2017**, 1124–1126.
31. Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [\[CrossRef\]](#)

32. Mesbahi, M.R.; Rahmani, A.M.; Hosseinzadeh, M. Reliability and high availability in cloud computing environments: A reference roadmap. *Hum.-Cent. Comput. Inf. Sci.* **2018**, *8*, 20. [[CrossRef](#)]
33. Hwang, Z. Design of Efficient Edge Computing based on Learning Factors Sharing with Cloud in a Smart Factory Domain. *J. Korea Inst. Inf. Commun. Eng.* **2017**, *21*, 2167–2175.
34. Min, H.; Jung, J.; Kim, B.; Heo, J. An Offloading Decision Scheme Considering the Scheduling Latency of the Cloud in Real-time Applications. *KIISE Trans. Comput. Pract.* **2017**, *23*, 392–396. [[CrossRef](#)]
35. Ra, J.-H. Offloading Decision Scheme Considering Lifespan in Mobile Cloud Computing. Master's Thesis, Chungnam National University, Daejeon, Korea, 2013. [[CrossRef](#)]
36. Zhang, K.; Mao, Y.; Leng, S.; Zhao, Q.; Li, L.; Peng, X.; Zhang, Y. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. *IEEE Access* **2016**, *4*, 5896–5907. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).