Received February 28, 2021, accepted April 7, 2021, date of publication April 9, 2021, date of current version April 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072164

Joint Resource Allocation Based on Traffic Flow Virtualization for Edge Computing

WON-SUK KIM[1], **(Member, IEEE), SANG-HWA CHUNG**[2], **(Member, IEEE), AND CHANG-WOO AHN**[2], **(Member, IEEE)**

[1]Department of Multimedia Engineering, Andong National University, Andong 36729, Republic of Korea
[2]Department of Computer Engineering, Pusan National University, Busan 46241, Republic of Korea

Corresponding author: Sang-Hwa Chung (shchung@pusan.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant by the Korean Government through [Ministry of Science and ICT (Information and Communications Technology) (MSIT)] under Grant NRF-2018R1D1A1B07049355 and Grant NRF-2020R1G1A1014096.

ABSTRACT Edge computing can provide network services with low latency and real-time processing by operating cloud services on network edges. Edge computing has numerous advantages such as low latency, local characteristics, and network congestion localization, however, associated resource management is becoming a significant challenge because of its features such as hierarchy, decentralization, and heterogeneity. Thus, a joint resource operation and management scheme for edge computing that can greatly reduce the traffic load in the network by considering the generated traffic load and computing resources is proposed in this paper. It is based on the concept of a virtual service flow (VSF) consisting of clients, data sources, a server instance, and external network entities for a specific service. The VSF consists of several drafts according to the expected server location, and each draft estimates a traffic load that can occur according to its characteristics. The location of the edge server of the VSF is determined for each VSF, and it is performed based on the VSF rejection and reconfiguration algorithm using the weighted vector bin packing algorithm that considers the client node coverage of the VSF. The proposed scheme is evaluated based on simulations that consider the actual characteristics of the network services and devices.

INDEX TERMS Cloud computing, edge computing, fog computing, heuristic algorithm, Internet of Things, network architecture, network management, software defined networking.

I. INTRODUCTION

Edge computing is a new computing model presented to address various issues such as high latency that can occur when current network services operate on cloud computing [1], [2]. It is a model that provides resources, such as computing, storage, and networks in the data center in the cloud computing model that is being provided, at the network edge close to the user. The network edge practically refers to hardware equipment, network switches, Wi-Fi access points, and mobile base stations that are geographically closer to the user [3], [4].

The main advantages of edge computing are low latency, location-oriented, mobility support, and device heterogeneity [5]. Edge computing can enhance real-time responsiveness compared to cloud computing, and can reflect regional characteristics at the network level, such as mobile base stations. Autonomous vehicles can be a good example of edge computing-based services. Edge computing may respond to requests for nearby traffic information or regional feature information from each vehicle only with information locally collected by the network [6]. In the case of an augmented reality (AR) navigation system, image processing such as object extraction and identification, and driver gaze tracking may be processed using computing resources of adjacent edge devices [7], [8].

Edge computing not only strengthens key indicators such as service responsiveness but also establishes itself as an enabler of specific services. Cloud gaming, which is in the spot-light as next-generation content, was proposed to overcome the limitations of client hardware. This operates in the form of collecting and processing user inputs received from a client on cloud servers and streaming the result of performing graphic rendering back to the client [9]. In such a service because users expect an immediate response to their input, response time becomes a crucial factor. Naturally, network

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

latency, which takes up most of the response time, becomes the most critical issue. In the existing environment based on cloud computing, even insensitive users may feel input delay, which soon results in reduced usability. Therefore, edge computing can be the core technology of cloud gaming [10].

In the case of cloud virtual reality (VR), network latency becomes even more critical. While latency in cloud gaming mainly affects the user experience such as operability, high latency in cloud VR causes high motion-to-photon (MTP) latency, leading to motion sickness [11]–[13]. As can be seen from these issues, edge computing can be a springboard for technological leaps in existing services such as autonomous driving, smart cities, and VR/AR. In addition, it will become an enabler of future content such as cloud gaming, cloud VR, and applications based on artificial intelligence (AI) [14]–[16].

A. MOTIVATIONS

Owing to the obvious advantages of edge computing, current mobile telecommunications standards support the concept in the form of multi-access edge computing (MEC). For edge computing to be readily utilized not only in this well-provisioned dedicated network but also in the general-purpose network system, there may be the following considerations:

First, edge devices that support edge computing in close proximity to users or things generally lack computing resources relative to physical machines in data centers. In terms of resource management, it is difficult to apply an efficient resource management scheme reflecting a global system perspective, such as resource virtualization. This is because the joint resource management of edge devices requires consideration of geographically distributed computing. Therefore, the operating location of the server instance of network services should be determined by considering the network topology as well as the computing resources of individual edge devices.

Second, it is necessary to consider different computing resource requirements depending on the service. Some services may require a great deal of processing and fewer storage requirements, and certain services may require more network resources than computing resources. In this context, if a specific server instance with a particularly high storage requirement is deployed on an edge device with a very small storage capacity and consumes the entire storage, other resources in the device may be wasted without being utilized. In addition, due to the rapidly changing characteristics of network services, rapid response to new resource factors that can be considered in the future is also required.

Third, edge computing should be sensitive to the traffic load within the network. The services such as cloud gaming, cloud VR, and augmented reality, which are next-generation content, may have a different level of traffic load from the previous ones [17], [18]. A service that utilizes edge computing can deploy its own server instance in the local network, which means that the traffic load in the local network will change significantly depending on the location of the server deployed on the edge device. Preventing congestion by reducing the traffic load through sophisticated server positioning will greatly contribute to performance evaluation, such as improving the low latency issue of edge computing.

Finally, in a network where edge computing is employed, most of the traffic will exist between the nodes connected to the clients and the node where the server instance of the service used by the client is located. It can be seen that a miniaturized server-client model is constructed within the network. To reduce the communication cost between the server and the clients, it is necessary to configure two or more server in-stances of a specific service as long as the service implementation allows.

B. CONTRIBUTIONS

In this paper, we propose a server instance operation and management scheme that considers various computing resource requirements of network services and device resource capacity, and simultaneously keeps the network congestion in the local network low. The proposed scheme creates a virtual service flow (VSF) and configures a server by estimating the traffic generated by each flow, and then deploys the servers to a network node in consideration of computing resources. Through this scheme, it is possible to flexibly respond to traffic inefficiency that may occur when edge computing is applied in various network situations such as various types of services, traffic patterns, and physical resource status.

The proposed scheme has the following contributions.

1) It contributes to a server placement technique that considers computing resources and network traffic together based on the usage of clients in the network to which edge computing is applied.
2) It is possible to increase the utilization of computing resources of edge devices by using a vector bin packing algorithm that considers multiple resources for server placement.
3) By estimating the generated traffic load before server placement, it is possible to provide a quick service response time and reduce network traffic.
4) This leads to a flexible traffic management environment by inducing the service administrator to configure multiple edge servers to improve the response time of the service within the same network.

II. RELATED WORK

Because the concept of edge computing was proposed with the spread of Internet of Things (IoT) services, research related to edge computing has been actively conducted in recent years. This concept, which is expressed in various forms such as fog computing and cloudlet, is moving towards a common goal of distributed processing and real-time support, but it means that there is no standardized or unified protocol until now. Paradoxically, this disproves that many parts of this field still exist as key research subjects. In the early days when the concept was presented, studies were mainly conducted on usability and application services, and in recent years, studies on computing and network resource management or system architecture have been actively conducted [19].

Hu *et al.* [20] implemented a facial recognition system using the fog computing concept. It analyzes the image transmitted from the fog node adjacent to the user and delivers the extracted face identifier to the cloud. It is stated that computing processes can be distributed from the cloud to fog nodes to enhance the overall processing efficiency and reduce network transmission. Xu *et al.* [21] presented a fog server deployment automation framework that analyzes the packets for a service request, retrieves the required application from the storage, and installs it on the fog node. In these studies, a fog device platform that supports various services was proposed. However, only a fog server for a single user connected to the fog device was taken into account, but computing resources were not considered.

Bellavista *et al.* [22] built real fog middleware using a message queuing telemetry transport (MQTT) protocol in a fog computing environment. By configuring the service in the form of a Docker container, the interoperability and portability of the service are facilitated. In addition, it enables the execution of multiple containers on a resource-limited node, such as Raspberry Pi, and has excellent scalability. Mahmud *et al.* [23] proposed a latency-aware application module management policy aimed at quality of service (QoS), application program optimization, and resource optimization, based on deadline time. In this study, a method of optimizing the number of resources without violating the QoS of the application was also investigated. This study explains that when a server instance of a specific fog node is relocated to another fog node, the corresponding node can be turned off. However, in general, the fog server operates on a network device, so it is not necessary to turn off the device even if there is no server in operation. Moreover, if the server is moved only by the power consumption problem of the device, the possibility of network congestion due to a service that generates a lot of traffic, such as cloud gaming, will increase significantly.

In the field of edge computing, researches on service implementation using edge computing architecture, edge nodes that operate deployed servers, and migration to apply edge computing have been conducted. In addition to these topics, a study regarding which server should be placed on which edge node for what purpose is also crucial from the network perspective.

Research on deploying virtual machines through vector bin packing algorithms in cloud data centers has also been actively conducted. Shi *et al.* [24] proposed a technique for placing virtual machines in physical machines using a vector bin packing algorithm in a data center. This scheme contributed to decreasing the power consumption of the data center by reducing the number of physical machines in which virtual machines are deployed through packing. Furthermore, a comparative study was conducted on various packing algorithms that can be used for virtual machine deployment. However, this has a limitation in that it cannot reflect various factors that can be considered in edge computing, and also, network topology is not considered. Edge nodes generally have fewer computing resources than physical machines in the data center, and recent network services utilize not only processors and memory, but also storage and graphic processing units. Therefore, these factors need to be considered.

Song *et al.* [25] studied the technique of arranging virtual machines in a data center by expanding the bin packing problem in multiple dimensions. The bin packing with variable item size algorithm was defined and compared with various types of algorithms. However, there is a clear difference between handling virtual machines within the data center and deploying server instances at the network edge in terms of the environment and the purpose of the algorithm.

The primary focus of deploying virtual machines in a data center is how few physical machines can handle the same resource requirements. The key to this operation is that the physical machine that is not running can be powered down to reduce power consumption. However, in an edge computing environment, how few nodes operate at the network edge, especially if the edge node is a network device, becomes a rather trivial matter. The focus of edge computing is how quickly it can respond to the requests of clients, sensors, and data sources. Handling the various factors in the network that affect the latency is more important than how few physical machines are used. To lower the latency, computing resources are readily considered, and simultaneously, several network factors are considered from hop count to network congestion control.

Mukherjee *et al.* [26], [27] described the fog computing instance operation method in terms of the latency experienced by the end-user. Certain end-user task with a tolerable latency is offloaded to the primary fog node in the primary coverage if the local system is unlikely to be able to process it within the deadline. By this process, the transmission latency for the task is sacrificed, but the overall latency can be decreased by reducing the computing latency with sufficient available computing resources. If it is determined that the task cannot be processed within the deadline with the available computing resources of the primary fog node, part of the task is offloaded to the cloud or secondary fog node at the expense of additional transmission latency. In this way, the latency experienced by the user is guaranteed as much as possible. This study has contributed to a more specific approach to computing resources such as CPU cycles. However, the proposed offloading scheme has a limitation of focusing only on the total latency of individual users.

In our previous study [28], the algorithm for deploying servers in an edge computing environment was presented. First, the server was temporarily deployed with a hop count between the client and the server instance as the main metric, and then the server with low priority was relocated heuristically using the vector bin packing algorithm. Conceptually, this study is an algorithm that efficiently combines computing

TABLE 1. Related work comparisons.

Refs.	Technique	Achievements	Limitations
Hu et al. [20]	Facial recognition system using the fog computing	Presenting an example of edge computing application	Only focuses on applications
Xu et al. [21]	Fog server deployment automation framework	Presenting a method of automatically deploying required edge servers based on service requests	Insufficient consideration for edge server deployment location
Bellavista et al. [22]	MQTT-based fog device middleware	Presenting an edge device implementation method using Docker	Insufficient consideration for edge server deployment location
Mahmud et al. [23]	Edge server placement	Presenting a resource optimization method based on application QoS	Focus on making edge nodes idle; Insufficient consideration for network
Shi et al. [24] Song et al. [25]	Virtual machine placement using VBP	Suggestion of VBP-based computing resource configuration method	Insufficient consideration for network
Mukherjee et al. [26][27]	Latency-aware task offloading	Suggestion of the latency-aware edge computing resource processing method	Consider the network only from the perspective of individual users
Lee et al. [28]	Edge server placement	Presenting an edge server deployment scheme based on network distance and VBP	Inflexible server operation and weak consideration of network traffic load

resources and network metrics, but there were several limitations in terms of practical applications.

First, the service flow presented in the study – a set of flows between multiple clients using a specific service and the edge server in the network that provides the service – excludes the concept of traffic load. When deploying the server instance for a specific service, only computing resources and hop counts were considered, but actual network usage such as traffic load was not taken into account. Using only the hop count as a network metric means that network congestion, such as a bottleneck of a specific path, cannot be prevented. Second, it is assumed that only one server of the service is in the network. The actual service can configure its own server in the form of multiple mirror servers, cache servers, and hierarchical servers, and the server deployment algorithm must explicitly consider these configuration types [29].

In this study, based on the concept of virtual network service flow, the edge computing server configuration and deployment algorithm that is close to practical applications is proposed. Furthermore, a heuristic technique for determining the optimal server operation type from the viewpoint of network congestion control is proposed.

III. SERVER MANAGEMENT SCHEME BASED ON VIRTUAL SERVICE FLOW

The network services based on edge computing process client requests through their server instances placed on the device, such as the network node or the host connected to the node in the edge network. The network configures server instances according to the role of the server by each service, determines the location of the server, and routes the client requests to the node where the server is placed. To examine the proposed scheme in detail, first, certain terms and components are defined.

First, the entity that operates a server instance at the network edge can be a piece of network equipment or a host connected in close proximity to it, and this is collectively referred to as an edge node. The server instance that responds to the client requests is called an edge server, and the server is placed and running on the edge node. The server operates on a single edge node. The edge server responds directly to client requests or with the results of communication with a cloud server or other network. The communication between the edge server and the cloud server can be regarded as communication between the gateway node and the edge node in which the edge server operates. In addition, it sometimes responds based on the results of communication with data sources such as sensors within the same network. When the network service communication form is abstracted from the edge computing model, it can be seen that the set of flows is formed between the nodes in which the edge server is running, the gateway node, the nodes to which the clients using the service are connected, and the nodes to which the data sources are connected. In this paper, this flow set is referred to as a virtual service flow (VSF).

Based on the VSF concept, the proposed algorithm places the edge server on the edge node, where the total of the traffic generated from the nodes constituting the VSF is estimated to be the lowest. In this process, the effect of the number and form of VSFs per service is analyzed, and the most appropriate number of VSFs and the server location of VSF are determined. In addition, when the server is determined in this manner, if the available resources of the target node are lesser than the required resources of the VSF servers, VSF relocation and reconfiguration are performed by using a modified vector bin packing scheme. As described above, the VSF operation and management scheme is proposed that minimizes the traffic load related to edge computing in the network and enables efficient use of computing resources of each node.

A. VIRTUAL SERVICE FLOW

VSF refers to a group of flows formed between the node to which the client is connected (the client node), the node on which the server responding to the request is located (the server node), the node to which other clients or data sources are connected (the association node), and the gateway node. In other words, it is a set of flows between a server node and a plurality of client nodes, flows between a server node and a plurality of association nodes, and flows between a server node and a gateway node. All related flows of a service can be represented by one VSF, which is called a global VSF. A global VSF includes nodes to which all requested clients are connected, their usage, the location of the association nodes, the computing load of the service according to the client usage, and the traffic load for each flow. Here, usage is a concept that can be expressed in various forms, for example, used time per day or traffic generated per unit time.

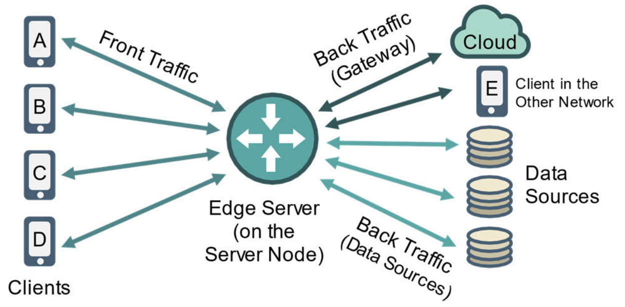

A
B
C
D
Clients
Front Traffic
Back Traffic (Gateway)
Cloud
E
Client in the Other Network
Data Sources
Edge Server (on the Server Node)
Back Traffic (Data Sources)

FIGURE 1. Various types of traffic and the components in the global virtual service flow for a certain service.

Fig. 1 conceptually shows the components and traffic types of the global VSF for a specific service. The traffic type can be classified as the traffic from each client node to the server node (FT; the front traffic), the traffic between the server node and the gateway node (BTG; the back traffic to the gateway), and the traffic between the server node and the association nodes (BTS; the back traffic to the data sources).

TABLE 2. Sample global virtual service flows.

id	Clients (Usage)	Data Sources	$R_{cpu}, R_{mem}, R_{sto}$	T_{ft}, T_{btg}, T_{bts}
F_1	3(2), 12(4)	D3, D5, D7	5/6/7	4/2/1
F_2	6(3), 7(1), 11(3)	D2, D4	6/6/12	1/0/4
F_3	9(2), 15(2), 16(3)	D1, D8	8/8/13	5/1/1

In the network topology shown in Fig. 2, suppose there are global VSFs, as shown in Table 2. In the case of F_1, when clients 3 and 12 use the service, the edge server communicates with the data sources D3, D5, D7, and the cloud server to process the client requests. R_{cpu}, R_{mem}, R_{sto} represent the computing resource requirements related to CPU, memory, and storage per usage, respectively. In order to handle client requests, the CPU, memory, and storage of the server node are required workloads by 5, 6, and 7, respectively, per usage. In addition, T_{ft}, T_{btg}, T_{bts} represent the factor of FT, BTG, and BTS per usage, respectively. In the example, 4 FT, 2 BTG,

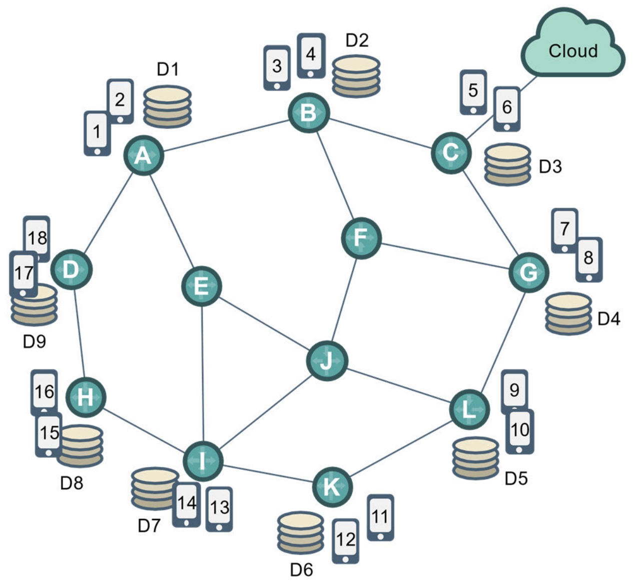


FIGURE 2. Sample network topology.

and 1 BTS are required per usage. This data can be obtained empirically within the edge node. A global VSF introduced in this subsection is a conceptual entity, and individual VSFs configured for each client node are introduced in Section 3.3.

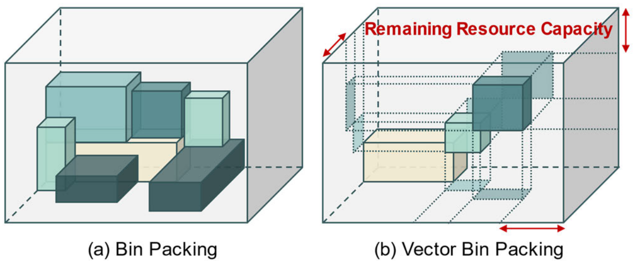


FIGURE 3. Differences between a common bin packing problem and vector bin packing problem utilized in this paper.

B. VECTOR BIN PACKING ALGORITHM

VSFs select the most suitable server node using the proposed algorithm. The computing resources of a single edge node are insufficient to host a large number of server instances at the same time. As a result, a certain node may not be able to operate all the servers of the VSF that have selected that node. In this case, servers with various requirements should be distributed across several nodes and deployed. This can be abstracted as a problem of properly arranging luggage of various sizes in bags with limited space; that is, it can be seen as the bin packing problem [30]. The general bin packing problem basically considers only the size of the luggage and bags, as shown in Fig. 3(a). Unfortunately, this concept cannot be applied directly because computing resources within a single node cannot be occupied simultaneously by multiple processes. In this study, the vector bin packing model is applied to handle the above-mentioned problem, as in the previous study [28].

Vector bin packing is a widely used solution when deploying virtual machines in cloud data centers [31], [32]. As shown in Fig. 3(b), each virtual machine occupies several types of resources it requires at the same time. The resources occupied by a specific virtual machine cannot be used by other virtual machines, and if a physical machine cannot satisfy any of the resources required by the virtual machine, the corresponding virtual machine is not deployed. For example, if a specific virtual machine with computing resource requirements very high for memory and close to 0 for other resource types is deployed, computing resources other than the memory of the physical machine may be wasted without being used. To resolve this problem, the goal may be to minimize wasted resources to place as many virtual machines as possible on the most appropriate physical server, and an approach for this is the vector bin packing model. In this study, physical machines are regarded as edge nodes, and virtual machines are considered as edge servers. As the vector bin packing model is applied, the available resources of the node and the requirements of the VSF servers are regarded as n-dimensional vectors. Then, by comparing the dot product of the node vector and the server vector, it is determined that the server with the smallest vector angle has the most appropriate resource requirement to be operated in that node.

The proposed algorithm is based on the vector angle comparison for each edge server at the edge node. In order for an edge node to operate as a VBP appliance, detailed specifications of the vector axes, that is, computing resources, are required. To focus on the description of the purpose of the proposed algorithm, it is assumed that computing resources are based on units called workloads [33]. The workload refers to the average amount of CPU, memory, and I/O work that a specific process uses during a specific time. For example, to represent the workload, under a specific implementation, CPU may use instructions per second (IPS), memory may use usage (megabytes), and storage can utilize input/output operations per second (IOPS) and usage (megabytes). Edge nodes can determine the size of their available computing resources from the kernel. The usage of each resource of the edge server can be measured before deployment by a service administrator, or can be acquired empirically during operation. Thus, the edge node can be used as a VBP appliance with a simple firmware implementation.

C. VSF-BASED TRAFFIC MANAGEMENT SCHEME FOR EDGE COMPUTING ENVIRONMENT

This section describes a technique for configuring service-specific VSFs for edge server placement in an edge computing environment and deploying edge servers based on VSFs in order to control traffic congestion. The scheme consists of three steps: configuring VSFs for each service, vector bin packing-based VSF server deployment, and VSF reconfiguration.

1) SERVICE SPECIFIC VSFS CONFIGURATION ALGORITHM

To perform the process of deploying edge servers based on the VSF, the VSF configuration for each service should be performed first. The VSF configuration generates VSFs and their drafts based on the information of client usages and data sources. The draft of VSF means a temporary plan generated for each candidate node where the VSF server can be deployed.

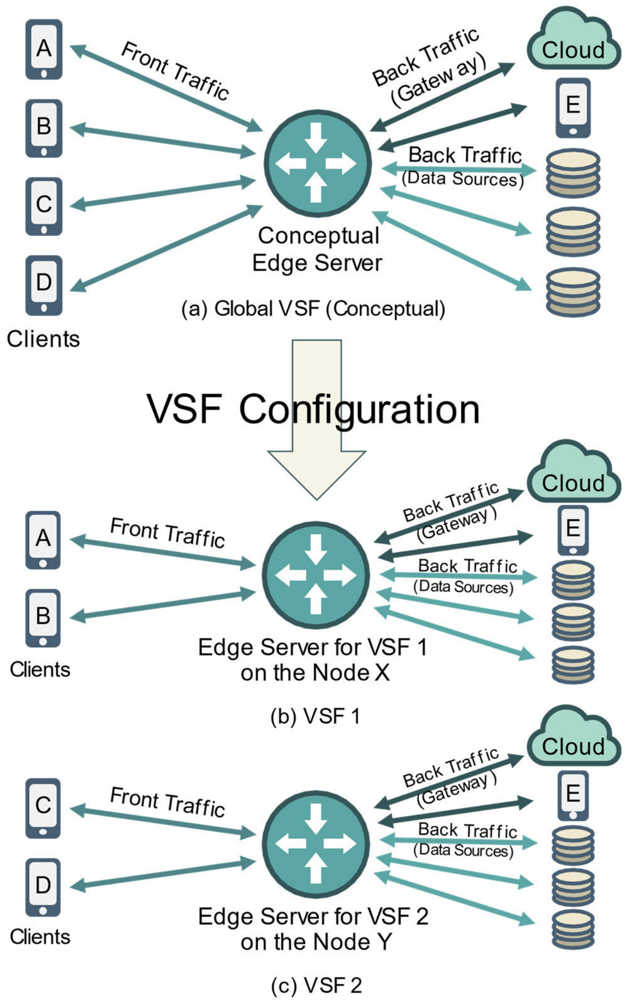


FIGURE 4. **Concept of virtual server flow configuration.**

Fig. 4 shows an example of the result of the VSF configuration. Fig. 4(a) shows the global VSF, which is conceptually present. Fig. 4(b) and 4(c) show an example of the global VSF divided into two VSFs. In the configuration step, as many VSFs as the number of clients are initially configured, but here it is assumed that two VSFs are configured for ease of description. According to the result, VSFs were configured such that clients A and B communicate with the edge server deployed on node X, and clients C and D communicate with another edge server deployed on node Y. Note that although multiple VSFs are configured, the server of each VSF still communicates with the cloud server and all data sources, as the global VSF. This can be seen as part of the abstraction of the process of collecting and processing data to respond to client requests. The back traffic load of each VSF is determined according to the usage of clients on the VSF. The proposed scheme does not actually configure the VSF for each service but configures the VSF for each client node.

Algorithm 1 Virtual Service Flow Configuration

```
1   $\mathcal{V} := \emptyset$ // the VSF set initialized as empty set
2   for each $s \in \mathcal{S}$ do
3       $\mathcal{C}_s \leftarrow$ the set of clients using the service $s$
4       $\mathcal{N}_c \leftarrow$ the set of nodes to which clients of $\mathcal{C}_s$
            are connected
5       for each $n \in \mathcal{N}_c$ do
6           $v_n \leftarrow$ the VSF for the client node $n$
7           $q_d^{v_n} \leftarrow$ the priority queue of drafts for
                VSF $v_n$, the priority is higher for the
                lower $\hat{\tau}$ of draft
8           for each $n_\sigma \in \mathcal{N}$ do
9               $\hat{\tau} \leftarrow GetETL(\mathcal{S}, n, n_\sigma)$ // estimated
                    traffic load on the shortest paths
                    between $n_\sigma$ and other nodes
10              $d_\sigma \leftarrow$ the VSF draft with the server
                    at $n_\sigma$, which includes $\hat{\tau}$ and path
                    information as meta data
11              $q_d^{v_n}.Push(d_\sigma)$
12          end for
13          $d_{ideal} \leftarrow q_d^{v_n}.Top()$
14          for each $d \in q_d$
                where $d^{\hat{\tau}} > \theta \times d_{ideal}^{\hat{\tau}}$ do
15              $q_d^{v_n}.Pop(d)$
16          end for
17          $\mathcal{V} := \mathcal{V} \cup \{v_n\}$
18      end for
19  end for
20
21  Output:
22  return $\mathcal{V}$
```

Algorithm 1 shows the algorithm for configuring the VSFs for each service. This algorithm is described using the example mentioned below. Fig. 2 shows how to configure VSFs and their drafts when the global VSF for each service is given, as shown in Table 2. In Algorithm 1, $\mathcal{S}$, $\mathcal{C}$, and $\mathcal{N}$ represent the set of network services, the set of clients, and the set of network nodes, respectively.

TABLE 3. Sample global virtual service flows.

$\mathcal{S}$	$\mathcal{C}_s$(usage)	$\mathcal{N}_c$(usage)	$\mathcal{N}_{assoc}$	$\mathcal{N}_{gw}$
s_1	3(2), 12(4)	B(2), K(4)	C, I, L	
s_2	6(3), 7(1), 11(3)	C(3), G(1), K(3)	B, G	C
s_3	9(2), 15(2), 16(3)	H(5), L(2)	A, H	

Table 3 shows the clients and their usage, client nodes, and association nodes for each service and the gateway node. Note that s_3 has three clients, but there are only two client nodes. Because clients 15 and 16 are connected to one node H, they are considered one client node in the VSF configuration process, and the usage of clients connected to the same node is summed. In other words, the VSF is configured based on the client node rather than the client or the service.

In Algorithm 1, from line 5, the configuration of the VSF for each client node is performed. Note that each VSF is not composed of single form but in the form of a priority queue for the drafts. Even if a specific VSF finds the most suitable server location for its client node at this step, it may not be possible to place the server on that node at a subsequent step. Therefore, drafts with the result of estimating the traffic load by regarding each node in the network as a server candidate are created in advance.

Algorithm 1-1 GetETL (Estimated Traffic Load)

```
1   Input:
2   $s \leftarrow$ the network service
3   $n_c \leftarrow$ the node which clients using the service $s$
        are connected
4   $n_\sigma \leftarrow$ the node where the server is to be located
5
6   $\hat{\tau}_{ft} \leftarrow T_{ft}^s \times usage_{n_c} \times dist(n_\sigma, n_c)$
7   $\hat{\tau}_{btg} \leftarrow T_{btg}^s \times usage_{n_c} \times dist(n_\sigma, n_{gw})$
8   $\hat{\tau}_{bts} \leftarrow T_{bts}^s \times usage_{n_c} \times \sum_{i=1}^{|\mathcal{N}_{assoc}|} dist(n_\sigma, n_{assoc}^i), \forall n_{assoc}^i \in \mathcal{N}_{assoc}$
9
10  Output:
11  return $\hat{\tau}_{ft} + \hat{\tau}_{btg} + \hat{\tau}_{bts}$
```

The process of creating drafts for each VSF is shown in lines 8 to 12. First, it is assumed that the VSF server for the client node n currently under consideration can be deployed on all nodes in the network. Assuming that the VSF server is deployed on a specific node, the traffic load of the VSF can be estimated based on the server node location. This assumption becomes one draft, and in the proposed algorithm, the draft list is generated by considering all nodes as VSF server node candidates. The GetETL function for calculating the estimated traffic load (ETL) $\hat{\tau}$ is shown in Algorithm 1–1. In line 6 of Algorithm 1–1, the estimated traffic load between the client node and the server node is calculated. This can be obtained by multiplying the client node usage $usage_{n_c}$ by T_{ft}^s, which is the FT load per usage of service s, and the distance of the shortest path between the client node and the server node. Similarly, the ETL between the gateway node and the server node and that between the association nodes and the server node is also calculated. In the above-mentioned case, it was assumed that communication with all association nodes should be performed to process the request of any client. Therefore, even if the ETL of a specific VSF is calculated, the traffic load to all association nodes must be calculated.

TABLE 4. Sample hop counts of shortest path.

$\mathcal{S}$	node type	node	Distances of shortest path											
			A	B	C	D	E	F	G	H	I	J	K	L
s_1	n_1	B	1	0	1	2	2	1	2	3	3	2	4	3
	n_2	K	3	4	3	3	2	3	2	2	1	2	0	1
	$\mathcal{N}_{assoc}$	C,I,L	7	7	6	9	6	6	5	8	6	5	5	4
s_2	n_1	C	2	1	0	3	3	2	1	4	4	3	3	2
	n_2	G	3	2	1	4	3	1	0	4	3	2	2	1
	n_3	K	3	4	3	3	2	3	2	2	1	2	0	1
	$\mathcal{N}_{assoc}$	B, G	4	2	2	6	5	2	2	7	6	4	6	4
s_3	n_1	H	2	3	4	1	2	3	4	0	1	2	2	3
	n_2	L	3	3	2	4	2	2	1	3	2	1	1	0
	$\mathcal{N}_{assoc}$	A, H	2	1	0	3	3	2	1	4	4	3	3	2
	$\mathcal{N}_{gw}$	C	2	1	0	3	3	2	1	4	4	3	3	2

TABLE 5. Sample estimated traffic load by server location.

$\mathcal{S}^*$	$\mathcal{N}_c$	$d^{\hat{\tau}}$ by n_σ											
		A	B	C	D	E	F	G	H	I	J	K	L
s_1	n_1	30	18	20	~~46~~	~~40~~	28	30	~~56~~	~~52~~	~~38~~	~~54~~	~~40~~
	n_2	~~92~~	~~100~~	72	~~108~~	80	88	60	~~96~~	72	76	44	48
s_2	n_1	~~54~~	27	24	~~81~~	~~69~~	30	27	~~96~~	~~84~~	~~57~~	~~81~~	~~54~~
	n_2	~~19~~	10	9	~~28~~	~~23~~	9	8	~~32~~	~~27~~	~~18~~	~~26~~	~~17~~
	n_3	57	36	33	~~81~~	~~66~~	33	30	~~90~~	~~75~~	54	~~72~~	51
s_3	n_1	~~70~~	~~100~~	~~130~~	50	~~80~~	~~110~~	~~140~~	30	60	~~85~~	~~90~~	~~115~~
	n_2	~~38~~	~~40~~	32	~~50~~	32	~~34~~	26	~~42~~	~~34~~	24	26	16

θ: 2

Tables 4 and 5 exhibit examples of the process of creating the draft list for each VSF in lines 8 to 12 of Algorithm 1. Table 4 shows the shortest path distance between certain nodes of each service and other nodes in the network. For example, the client node n_1 of s_1 is in node B, and the shortest distance between this node and node A is 1, and between this node and node H is 3. Note that a VSF should communicate with all association nodes to respond to any client, so the distance of the association nodes is the sum of the distances of the elements. For example, the elements of the association nodes of s_1 are nodes C, I, and L, and when the server node is A, the total shortest distance to reach all association nodes is 7. The gateway node is C, and the shortest distance to other nodes in the network is also indicated.

Table 5 shows the ETL for each draft according to the server location of each VSF, after configuring the VSF for each client node of the service by Algorithm 1. According to the table, in the case of the draft in which the VSF server for the node n_1 of s_1 is deployed in node D, the total traffic load on the path used by the VSF in the network is estimated to be 46. It is calculated using Algorithm 1–1. The usage of the client node n_1 of s_1 is 2, T^s_{ft} is 4, and the shortest distance between the client node and the server node is 2, hence the total amount of traffic load on the path is $2 \times 4 \times 2 = 16$. In this way, the total traffic load on the path between the server node and the association nodes can be estimated as $2 \times 1 \times 9 = 18$, and the total traffic between the server node and the gateway node is $2 \times 2 \times 3 = 12$. Through the summation of these results, the ETL is calculated when the VSF server is located at node D. When a VSF for each client node is configured through this process, the ETL for each server location candidate of the VSFs can be obtained.

While calculating the ETL of drafts for each VSF, the priority queue $q^{v_n}_d$ as a container of the drafts determines the priority based on the ETL of the draft. The lower the estimated traffic of the draft, the higher the priority. As in line 13, the draft with the lowest ETL becomes the ideal draft d_{ideal}. Lines 14 to 16 show the draft validation process in which drafts of lower quality among drafts of the same VSF are removed from the queue. In the proposed algorithm, the low-quality draft means a draft with ETL that exceeds θ times the ETL of the ideal draft. That is, even though the ideal draft cannot be selected in the subsequent VSF server deployment process, the traffic load is reduced by excluding drafts that are expected to have excessively high traffic load from the next best option.

TABLE 6. Initial VSF drafts for each client node.

$\mathcal{V}$	$\mathcal{S}$	$\mathcal{N}_c$	q_d, expressed in ($\mathcal{N}_\sigma, \hat{\tau}$)
v_1	s_1	n_1	(B,18), (C,20), (F,28), (A,30), (G,30)
v_2	s_1	n_2	(K,44), (L,48), (G,60), (C,72), (I,72), (J,76), (E,80), (F,88)
v_3	s_2	n_1	(C,24), (B,27), (G,27), (F,30)
v_4	s_2	n_2	(G,8), (C,9), (F,9), (B,10)
v_5	s_2	n_3	(G,30), (C,33), (F,33), (B,36), (L,51), (J,54), (A,57)
v_6	s_3	n_1	(H,30), (D,50), (I,60)
v_7	s_3	n_2	(L,16), (J,24), (G,26), (K,26), (C,32), (E,32)

Table 6 shows the VSF configuration results for each client node. The VSF consists of several drafts, as shown. In the table, a draft is expressed as a pair of expected server nodes and ETL. For example, the VSF for the client node n_1 of s_1 consists of five drafts, and the draft in which the server is in node B is the ideal draft. In Table 5, ETL was calculated by considering all nodes in the network as server deployment candidates, but candidates with double or more ETL compared to the ETL of the ideal draft were not included in the draft list.

In the process of VSF configuration for each service, VSFs are configured for each client node, and drafts for each VSF are created. To restate the proposed scheme, the scheme configures VSFs not for each service, but for each client node. The former method is similar to that proposed in a previous study. The advantages and disadvantages of these two approaches can be compared without considering detailed factors, such as synchronization between VSFs or shared computing resources. When VSFs are created for each service, the number of generated VSFs decreases. That is, if the network size to be considered is expanded, the complexity of VSF routing or managing computing and network resources is reduced. However, if VSF is created for each client node, the management complexity is slightly increased, but the traffic load can be finely controlled. It is judged that problems caused by management complexity can be helped by software-defined networking (SDN) and container technology [34], [35]. Therefore, the proposed scheme selected to operate the VSF for each client node in order to concentrate on traffic load control.

2) VSF DRAFT SELECTION ALGORITHM

This section describes the second step algorithm for selecting the most suitable draft from the draft list for each VSF. Selecting a draft means that the server location of the VSF will be determined to be of the corresponding draft. After drafts of all VSFs are selected by this algorithm, the server location of the draft for each VSF is applied to the network.

Algorithm 2 shows an algorithm for selecting the draft for all VSFs. First, as shown in lines 5 to 8, the ideal drafts of each VSF are added to the final plan $\mathcal{P}$ in order to initialize the plan. At this point, $\mathcal{P}$ is the set of selected drafts, which is planned to be applied to the network.

For the drafts to be applied, all nodes that are the target of deploying server nodes of drafts currently under consideration should check whether their available computing resources can satisfy the requirements of the VSF servers that have selected them. In lines 11 to 17, for all nodes in the network, the required computing resource loads of all servers scheduled to be deployed on that node are summed by resource type and compared with the resource capacity of the node. At this time, r_{req}^i is the required i-th resource, and r_n^i is the remaining capacity of n for the i-th resource. If a specific node cannot satisfy even one kind of resource, the node becomes an overloaded node.

In the overloaded node, the server rejection process is executed, and at this time, the vector bin packing (VBP) algorithm mentioned above is modified for the proposed scheme and employed. When calculating VBP, a vector with a dimension as many as the number of resources considered in the current system is used. Lines 21 to 29 show the selection process of the rejected VSF. The vector angle of the server is obtained through the dot product operation between the requested resource vectors of all draft servers that were selected to be placed in the overloaded node and the available resource vectors of the node. The fact that the vector angle is larger than that of other servers increases the probability that there are resources that can be wasted when the server is deployed. Contrastingly, as the vector angle approaches 0, it indicates that the ratio of the resources of the server and the node becomes similar, meaning that unused resources can be minimized. That is, the VSF that owns the draft that intends to deploy the server with the highest vector angle becomes the rejected VSF. The rejected VSF discards the currently selected draft, selects the draft of the next priority, and performs Algorithm 2 again.

As mentioned above, in the proposed scheme, VBP is modified appropriately and utilized. When calculating the

Algorithm 2 VSF Draft Selection

```
1  Input:
2  $\mathcal{V} \leftarrow$ the set of VSFs
3
4  $\mathcal{P} \leftarrow$ the set of selected VSF drafts as the final plan
5  for each $v \in \mathcal{V}$ do
6      $d_{ideal} \leftarrow q_d^v.Pop()$
7      $\mathcal{P} := \mathcal{P} \cup \{d_{ideal}\}$
8  end for
9  while $true$ then
10     $n_o \leftarrow null$// the overloaded node
11     for each $n \in \mathcal{N}$ do
12         $\left(r_{req}^1, r_{req}^2, r_{req}^3\right) \leftarrow GetReqRsrc(\mathcal{P}, n)$
13         if $r_n^1 < r_{req}^1 \vee r_n^2 < r_{req}^2 \vee r_n^3 < r_{req}^3$ then
14             $n_o \leftarrow n$
15             break
16         end if
17     end for
18     if $n_o = null$ then break
19     $a_{max} \leftarrow -1$
20     $\vec{r_n} \leftarrow \left(r_{n_o}^1, r_{n_o}^2, r_{n_o}^3\right)$
21     for each $\sigma \in n_o^{planned\ VSF\ servers}$ do
22         $\vec{r_\sigma} \leftarrow \left(r_\sigma^1, r_\sigma^2, r_\sigma^3\right)$
23         $\omega \leftarrow \left(|\mathcal{N}_c| - \left|\mathcal{N}_c^\sigma\right|\right) / |\mathcal{N}_c|$, $\mathcal{N}_c$ is the client
               node of the service in VSF to which
               $\sigma$ belongs
24         $a \leftarrow \cos^{-1}\left(\vec{r_n} \cdot \vec{r_\sigma} / |\vec{r_n}|\,|\vec{r_\sigma}|\right) \times \omega$
25         if $a_{max} < a$ then
26             $a_{max} \leftarrow a$
27             $v_{reject} \leftarrow v_\sigma$
28         end if
29     end for
30     if $\sim \left(q_d^{v_{reject}}.IsEmpty()\right)$ then
31         $\mathcal{P} := \mathcal{P} - \left\{d\ of\ v_{reject}\right\}$
32         $\mathcal{P} := \mathcal{P} \cup \left\{q_d^{v_{reject}}.Pop()\right\}$
33     else
34         call $ReconfVSF\left(v_{reject}\right)$
35         exit with error
36     end if
37 end while
38
39 Output:
40 return $\mathcal{P}$
```

vector angle of the server, the vector angle with the node is not just compared, but the value obtained by applying the weight of the VSF to the vector angle is considered as a metric. The weight ω is the ratio of the number of client nodes covered by the VSF among the number of client nodes of the service. The VSF initially covers only one client node but can be responsible for more than one client node in the VSF reconfiguration process. In general, as the number of client nodes covered by one VSF increases, the efficiency of traffic control by the proposed scheme decreases. For example, in Table 5, if n_1 and n_2 of s_1 are configured to belong to separate VSFs, the sum of the ETL of the ideal draft is 62. However, if the two client nodes are configured to belong to the same VSF, the ETL of the ideal draft is 90 when the server is in node G. That is, if one VSF covers more client nodes, it is difficult to maximize the gain due to the server location, and traffic control efficiency decreases.

In VBP, the VSF with the server with a large vector angle is rejected. The higher the ratio of the client nodes covered by the VSF, the lower the weight. That is, if the ratio of the covered client nodes is high, a low weight is applied and the probability of becoming the rejected VSF decreases. The more the VSF is reconfigured, the ETL efficiency of subsequent drafts can be further reduced, so the probability of rejection of that VSF is lower. In addition, the VSF that is already in charge of all client nodes and can no longer be reconfigured is prevented from being rejected by setting the metric to 0. For example, if the VSF of service A with five client nodes covers two client nodes, then $\omega_A = (5-2)/5 = 0.6$, and the VSF of service B with four client nodes covers 1, then $\omega_B = (4-1)/4 = 0.75$. If the two VSF servers have the same vector angle with the node, the VSF of service B is rejected.

In line 30, the operation branches depending on whether there is a remaining draft of the rejected VSF. If there is a draft in the VSF, the current draft of the VSF is removed from $\mathcal{P}$, and the draft of the next priority is added, and then the VSF server deployment validation on $\mathcal{P}$ is performed again. If there is no draft left in the VSF, the VSF reconfiguration should be performed. Algorithm 2 is then terminated, and after VSF reconfiguration is performed, this algorithm is performed again from the beginning.

3) VSF RECONFIGURATION ALGORITHM

If the drafts of a specific VSF of the service are exhausted, the corresponding VSF needs to be reconfigured. In Algorithm 1, only the drafts with ETL less than θ times of the ideal draft were maintained. Suppose that low-quality drafts are not being removed. When n_1 and n_2 of s_1 in Table 5 are configured to belong to separate VSFs and only the VSF covering n_2 is continuously rejected and the last draft is selected, the sum of the ETL of the VSFs of n_1 and n_2 becomes $18+108+126$. However, if the two nodes are reconfigured to belong to the same VSF, the ETL of the ideal draft is 90, when the server of the VSF is in G. Instead of selecting the next draft by using all drafts created during the VSF configuration process, the traffic efficiency can be enhanced by configuring a new VSF covering two or more client nodes at a specific point in time.

Algorithm 3 shows the VSF reconfiguration algorithm. In line 7, a temporary VSF is configured to include both the client node(s) of the rejected VSF and the client node(s) of another VSF in the service to which the rejected VSF belongs. As in Algorithm 1, drafts are generated assuming that the server is in each node in the network, and then the ETL of each draft is calculated. ETL of the VSF covering two or more client nodes adds up the results of Algorithm 1–1 for each client node. When all drafts are determined, the ideal draft is selected and drafts with an ETL exceeding double that of the ideal draft are removed. Then, the average ETL of all drafts $\bar{\tau}$ was calculated. After finding $\bar{\tau}$ for all temporary VSFs, the VSF with the lowest $\bar{\tau}$ becomes the newly reconfigured VSF. When calculating the reconfigured VSF, the average ETL rather than ETL of the ideal draft is employed because the drafts may be discarded in order in Algorithm 2. The VSF that formed the new VSF with the rejected VSF is removed from the VSF set $\mathcal{V}$, and the new VSF is added to the VSF set.

Table 7 shows the steps in the reconfiguration process when the draft of VSF v_4 responsible for n_2 of s_2 is exhausted. The temporary VSF v^1_{temp} covers n_1 and n_2, and the ideal draft is when the server is located in node C. In addition to the ideal

Algorithm 3 VSF Reconfiguration

```
1  **Input:**
2  $v_{reject} \leftarrow$ the rejected VSF
3
4  $s \leftarrow$ the service to which $v_{reject}$ belongs
5  $\mathcal{V}_s \leftarrow$ the set of VSFs of the service $s$
6  **for each** $v \in \mathcal{V}_s$ **where** $v \neq v_{reject}$ **do**
7      $v_{temp} \leftarrow$ the VSF configured temporarily to
             cover both the client nodes of $v$ and
             of $v_{reject}$
8      $\tau_{min} \leftarrow \infty$
9      **for each** $n_\sigma \in \mathcal{N}$ **do**
10         $\hat{\tau}_{temp} \leftarrow$ set as the estimated traffic load for
               $v_{temp}$, which is the sum of all the
               covered client nodes specific ETL
               when the server is located in $n_\sigma$
11         $d_\sigma \leftarrow$ the draft of $v_{temp}$ when the server is
               in $n_\sigma$, which includes $\hat{\tau}_{temp}$ and path
               information as meta data
12         $q_d^{v_{temp}}.Push\,(d_\sigma)$
13     **end for**
14     Remove any drafts with ETLs that exceed twice
           the ETL of ideal draft
15     $\bar{\tau} \leftarrow$ set as the average value of ETL for all drafts
           included in $q_d^{v_{temp}}$
16     **if** $\tau_{min} > \bar{\tau}$ **then**
17         $\tau_{min} \leftarrow \bar{\tau}$
18         $v_{new} \leftarrow v_{temp}$
19         $v_{removed} \leftarrow v$
20     **end if**
21 **end for**
22 $\mathcal{V} := \mathcal{V} - \{v_{reject}, v_{removed}\}$
23 $\mathcal{V} := \mathcal{V} \cup \{v_{new}\}$
24
25 **Output:**
26 **return** $\mathcal{V}$
```

TABLE 7. **Example of VSF reconfiguration for service 2.**

$\mathcal{V}$	$\mathcal{N}_c$	$d^{\hat{\tau}}$ by n_σ											
		A	B	C	D	E	F	G	H	I	J	K	L
v_3	n_1	~~54~~	27	24	~~81~~	~~69~~	30	27	~~96~~	~~84~~	~~57~~	~~81~~	~~54~~
v_4	n_2	~~19~~	10	9	~~28~~	~~23~~	9	8	~~32~~	~~27~~	~~18~~	~~26~~	~~17~~
v_5	n_3	57	36	33	~~81~~	~~66~~	33	30	~~90~~	~~75~~	54	~~72~~	51
v^1_{temp}	n_1, n_2	~~73~~	37	33	~~109~~	~~92~~	39	35	~~128~~	~~111~~	~~75~~	~~107~~	~~71~~
v_5	n_3	57	36	33	~~81~~	~~66~~	33	30	~~90~~	~~75~~	54	~~72~~	51
v_3	n_1	~~54~~	27	24	~~81~~	~~69~~	30	27	~~96~~	~~84~~	~~57~~	~~81~~	~~54~~
v^2_{temp}	n_2, n_3	76	46	42	~~109~~	~~89~~	42	38	~~122~~	~~102~~	72	~~98~~	68

draft, the valid drafts are when the server is located in nodes G, B, and F, and the average ETL of v^1_{temp} is 36. The other temporary VSF v^2_{temp} is responsible for n_2 and n_3, and the ideal draft is when the server is located in node G. The valid drafts are when the server is in nodes C, F, B, L, J, and A, and the average ETL is 54.86. Thus, v^1_{temp} becomes the new VSF and is added to the VSF set, and v_3 and v_4 are removed from the set.

Subsequent to the execution of Algorithm 3, Algorithm 2 is performed again with the newly configured $\mathcal{V}$. The VSF rejected while performing Algorithm 2 selects the next draft, and if there is no next draft, the VSF is reconfigured in Algorithm 3 again. In Algorithm 3, if there is no VSF other than the rejected VSF in the service, that is, if the rejected VSF covers all the client nodes of the service, the algorithm finally fails. Because the failure of the algorithm is directly related to the reliability, the weights were used to prevent frequent failures. Through this process, if all VSF servers are expected to be deployed so as not to exceed the computing resource load on all nodes, the proposed scheme returns the final plan $\mathcal{P}$. Based on $\mathcal{P}$, the accompanying operations, such as server deployment, routing, and live migration, are performed.

The time complexity from VSF configuration to final deployment plan establishment is as follows. In the proposed scheme, VSF configuration is first completed, and then Algorithms 2 and 3 are repeatedly performed to establish the final plan. Since VSF configuration generates VSFs for each edge node for each client node for each service, the time complexity is $O\left(|\mathcal{S}|\,|\mathcal{N}|^2\right)$. The time complexity of the final plan establishment is $O\left(|\mathcal{V}|\,|\mathcal{N}|\cdot|\mathcal{N}|^2\right)$ because the VSF is reconfigured when the draft of a rejected VSF is exhausted while processing the VSFs. The latter is the actual time complexity of the proposed algorithm. The first term of complexity indicates the operation for drafts of all VSFs, and the latter term indicates the VSF reconfiguration. The number of VSFs $|\mathcal{V}|$ is the sum of the number of client nodes for each service, and the number of client nodes of the corresponding service also affects the VSF reconfiguration. Thus, as the number of client nodes of each service decreases, the complexity greatly decreases. The complexity can be significantly reduced in an actual scenario, because it can be rare for all clients located on all nodes use all services. In addition, the draft validation also reduced the complexity of final plan establishment. The complexity of the linear programming has been studied so far, but it remains at a considerably higher level than the proposed algorithm, such as having exponential complexity under normal circumstances [36].

IV. SIMULATION RESULTS

In this paper, a virtual service flow-based traffic control management scheme is proposed. The proposed scheme can effectively control traffic congestion in the edge network by reflecting features such as resource constraints of edge nodes while considering various requirements for computing resources and traffic of services in an edge computing environment. VSFs and their drafts are created based on the client nodes of the services, and a method of selecting another draft or reconfiguring the VSF according to whether the computing resources of the nodes are satisfied or not is proposed. This secures the efficiency of managing computing resources in the network and reduces the network traffic load. We evaluate the effectiveness of the proposed scheme from various perspectives through a comparison with existing methods using simulation.

A. SIMULATION ENVIRONMENT

The network topology for the simulation is given in Fig. 2. The locations of clients and data sources are as shown in Fig. 2, unless otherwise stated. The usages per client and service characteristics change according to the purpose of the simulation. Each network node utilized as an edge node has its own capacity for computing resources. It is assumed that there are several types of services in the network, and each service provides edge services by operating as many VSFs as the maximum number of client nodes. The computing resource load and traffic load according to client usage are determined within a specific range, and this is information that can be empirically secured by system monitoring.

The simulation parameters are as follows: The values mentioned below are applied by default for each simulation unless otherwise specified. The figures presented assume a situation where the usage is more concentrated or an edge node with quite poor computing power compared to the actual environment. That is, it is assumed that VSF rejection and reconfiguration can occur frequently. The available resources of the edge node are determined within the range of $r^1_n \in [100, 500]$, $r^2_n \in [100, 1000]$, $r^3_n \in [100, 3000]$, and the unit may be workload. Resources 1, 2, and 3 refer to processor, memory, and storage, respectively, in this simulation. The resource and traffic load per usage for each service is determined within the range of $r_1 \in [0, 5]$, $r_2 \in [0, 10]$, $r_3 \in [0, 50]$, $T_c, T_g, T_s \in [0, 5]$, and the unit can be workload and Mbps. There are 10 types of services within the network, each of which is used by up to five clients and has up to two data sources. The threshold value θ related to the tolerance value of valid drafts was set to 2, and at least three drafts were left to prevent too frequent failures. There is only one gateway node in the network, and it is set as a random node for each simulation. The usage of each client was determined to be between 0 and 10.

The proposed algorithm is compared with the first fit (FF), first fit decreasing (FFD), best fit decreasing (BFD), FFD by traffic (FFDt), and the scheme proposed in our previous paper (Single VSF Only; SVSF). In addition, it contributes to identifying the appropriate algorithm for each network situation through simulation by varying the weight of the proposed algorithm. FF composes VSFs for each client node of the service but arranges VSFs in random order from the first node that can be deployed. FFD is similar to FF, but VSFs are arranged after sorting in descending order by the required resource size. The required resource size is the sum of the three types of computing resource load. BFD is similar to FFD, but the VSF server is deployed on the node that is expected to have the least remaining resources. FFDt is similar to FFD, but sorts VSFs in descending order by the expected traffic of VSF, not by the required resource size. The expected traffic is a value obtained by multiplying the usage of the client node by the sum of the traffic factor per usage of each traffic type, excluding the number of paths used by the VSF. The node order mentioned here was rearranged in random order for each simulation.

The reasons for selecting somewhat classical FFD, BFD, etc. as the target scheme to be compared are as follows. The proposed scheme configures VSFs in a preprocessing step and establishes a deployment plan based on the configured VSFs. Since the proposed scheme requires a preprocessing step, it is unreasonable to be directly compared with other methods. Thus, classical solutions have been chosen to compare the effectiveness of final planning in situations where VSFs are already configured. The comparison target schemes are somewhat traditional, but have been effectively modified for simulation. This will sufficiently verify the performance of the preprocessing and deployment algorithm of the proposed scheme.

In our previous work, all the client nodes of the service consisted of one VSF. The VSF drafts were sorted in ascending order based on the number of hops. The VSFs were sorted in ascending order with the number of drafts having the same number of hops as the ideal draft, and then deployed in the sorted order. At the time of deployment, if the ideal draft cannot be deployed in a node, the next draft is selected, and if there is no selectable draft, the algorithm fails.

The proposed scheme has four weight variations: w0, w1, w2, and nRcf. The w0 does not use any weight; that is, the rejected VSF is selected regardless of the number of client nodes covered by the VSF. The w1 is the same as that of the proposed algorithm. The w2 considers the influence of the number of covered client nodes of the VSF to be doubled when calculating the weight. That is, the weight is calculated as $\omega \leftarrow \left(|\mathcal{N}_c| - 2\left|\mathcal{N}_c^{\sigma}\right|\right) / |\mathcal{N}_c|$. The nRcf does not perform a VSF reconfiguration and only calculates weights. In other words, the rejected VSF is not reconfigured as the new VSF integrated with other VSF in the service, but only reduces the weight of the corresponding VSF by half, and then performs Algorithm 2 again from the beginning. If the same VSF is rejected 4 times, the algorithm is considered to have failed.

Factors that change with each simulation include not only the main simulation variables, but also the computing resource of the nodes, the service resource factor per usage, the service popularity, the location of clients, and the location of the gateway node. To remove the distortion caused by the changing factors, all simulations were performed 100,000 times, and all results were expressed using an arithmetic mean value.

In this simulation, all random variables follow a uniform distribution. The changing factors may follow a normal distribution, Zipf distribution, and sometimes Pareto distribution, depending on their characteristics in an actual scenario [37], [38]. However, even if individual changing factors follow a separate distribution, a large number of iterative simulations are sufficient to attenuate the effects of the separately applied distribution. For example, assume that service usage and popularity follow the Zipf distribution. This becomes similar results to the case where only some popular services exist by the number of simulation iterations. The remaining minor services only have a negligible effect on the performance trend according to the simulation variable. Therefore, it can be said that the application of the uniform distribution to the environment variables is appropriate for analyzing the effects of the simulation variables.

B. SIMULATION RESULTS AND DISCUSSION

The simulation aims to evaluate the total traffic load, average computing resource utilization, and reliability of the algorithm for various situations in the network. In addition to comparison with other schemes, the results according to the VSF rejection weight are also compared.

Because the simulation is performed assuming an environment where the amount of usage is concentrated, a certain deployment scheme may fail depending on random parameter values. Failure means that a specific VSF server cannot be deployed in a given environment, which can be expressed as an indicator of the algorithm success rate. Although the success rate is directly related to reliability, it is not fatal because alternate algorithms exist in case of failure. However, it is related to deployment efficiency. For example, increasing the tolerance threshold in the draft validation stage increases the success rate but increases the traffic load. This part can be regarded as an optimization factor in actual implementation.

The simulation parameters consist of various elements, such as the number of services and the number of clients per service, some of which are directly related to the total usage of all clients. In the simulation, the number of clients was used as a concept that depends on the number of services rather than an independent parameter. As the usage range for each client is specified, the total usage of clients increases as the number of services increases. Likewise, when the number of clients per service increases, the total usage increases. This is intentional and changing only the number of clients

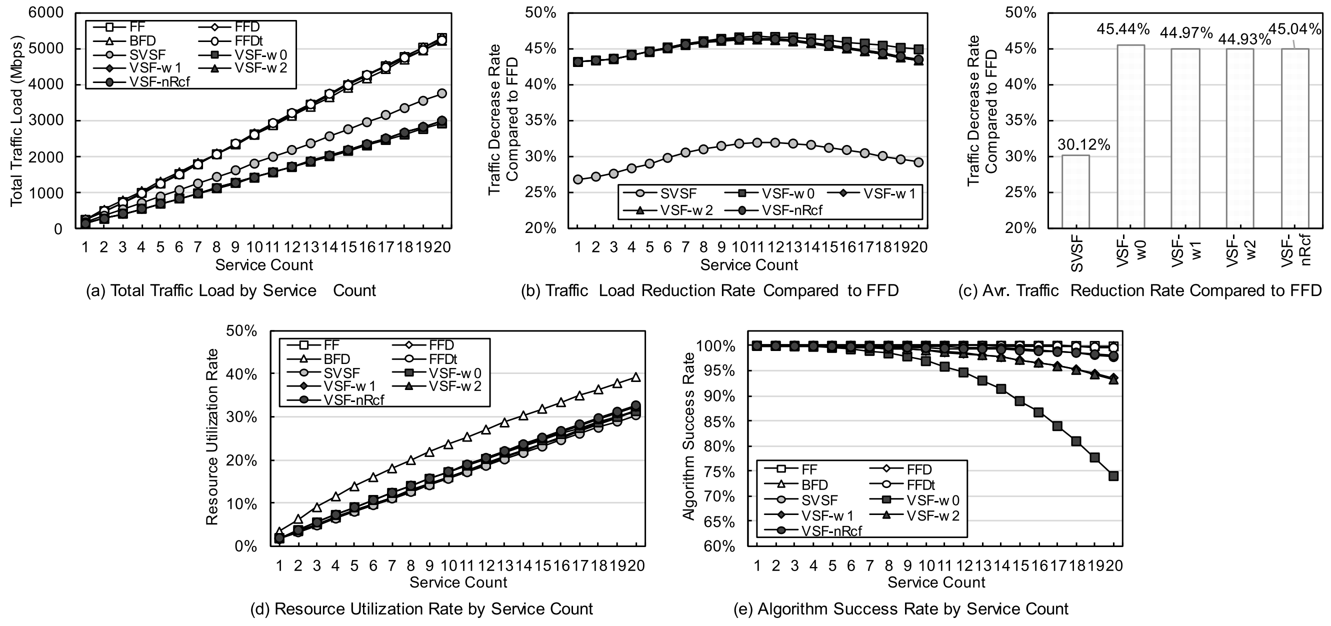


FIGURE 5. Simulation results in various aspects of the network by the number of services.

or services while limiting the total usage is because the impact on the network is negligible, except for some special circumstances. Some of those situations do not significantly affect the results of iterative simulations assuming random situations, making the reason to perform the simulation less.

First, a simulation for the number of services was performed to evaluate the effect of service diversity on the network to which the algorithm was applied. Fig. 5 shows the simulation results of various aspects as the number of services changes from 1 to 20. The other parameters are the same as the aforementioned values. For example, when there is one service, the number of clients is 1 to 5, and when there is 20, a total of 20 to 100 clients can exist. Fig. 5(a) shows the total traffic load in the network according to the number of services. As mentioned above, it can be seen that the number of services and the total traffic load in the network are in direct proportion. However, in each situation, the total traffic load for each algorithm shows different patterns.

To examine this in detail, Fig. 5(b) shows the traffic load reduction rate of other algorithms compared to the FFD scheme according to the number of services. Only major algorithms such as SVSF appear here, including variations of the proposed algorithm. When the number of services is approximately 10 to 12, most algorithms show the best performance compared to FFD, and other cases show similar results numerically. However, as the number of services increases to more than 20, the reduction rate compared to FFD is expected to decrease, because the success rate of the major algorithms decreases as the usage approaches saturation. That is, in a situation where the amount of computing resource requirements is large and the remaining resources of the node are insufficient, individually responding to the VSF deployment increases the possibility that the placement of the lowest priority VSF is continuously impossible. The low-priority VSF means a VSF having a large vector angle with server nodes of the ideal or semi-ideal draft. In this case, the algorithm will fail, and hence, the frequency of using an alternate algorithm, such as FFD and FFDt, increases.

Fig. 5(c) shows the average performance improvement of the major algorithms compared to FFD. The SVSF algorithm proposed in the previous paper reduced the traffic load by 30.12% compared to FFD, and the VSF algorithms proposed in this paper reduced the traffic load by an average of 45.09%. The variations of the proposed algorithm showed comparable performance improvement.

If the first-fit algorithm is performed after sorting based on the ideal draft ETL for each VSF, the success rate may increase slightly, and the traffic may decrease slightly. However, users of services with low ETL and high resource usage may experience QoS degradation due to high latency. In this case, the QoS imbalance between services increases.

Fig. 5(d) shows the resource utilization rate for each algorithm according to the number of services. The resource utilization rate is an arithmetic average of the utilization rate by resource type of all nodes. The higher this value, the lower the resource utilization efficiency, which means that more VSF servers are deployed where the available resources are low. In addition, it also implies a situation in which servers of other VSFs cannot be deployed to the node even though other resources of the node remain because a specific resource is saturated. Because each simulation is performed for the same amount of usage, the case in which the resource utilization rate is higher than that of other algorithms can be seen in the following example. Suppose that node A has

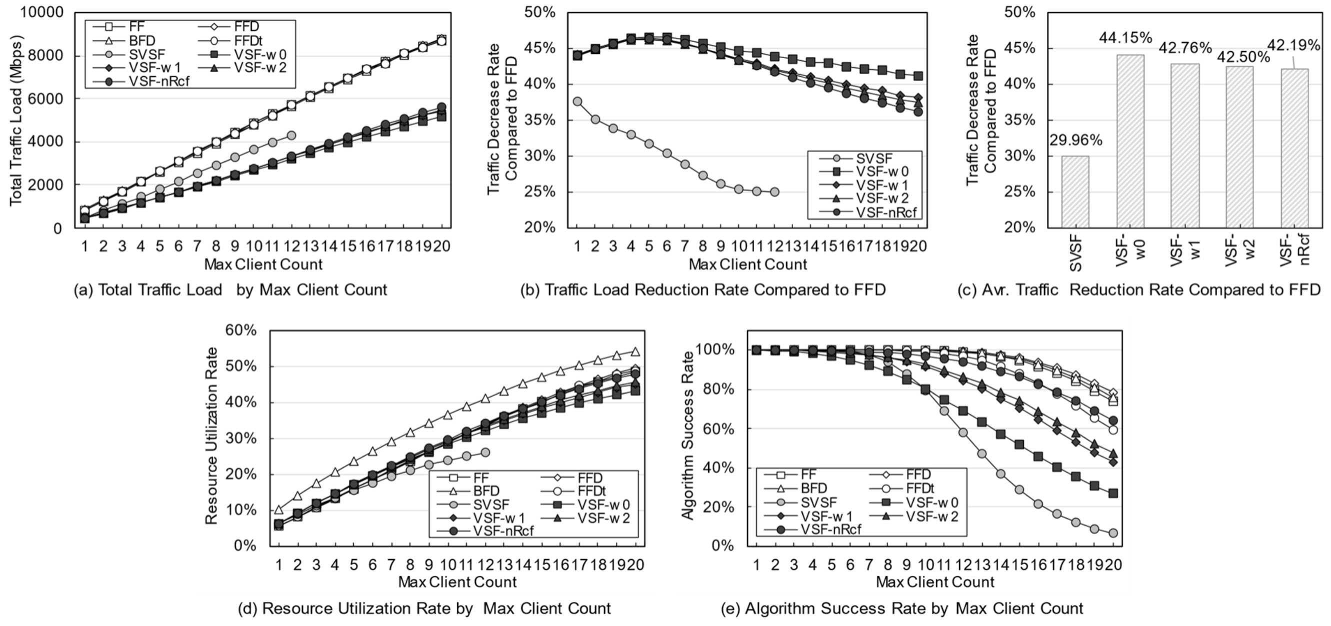


FIGURE 6. **Simulation results by the maximum number of clients per the service.**

(10, 10, 20) available resources, and node B has (20, 20, 20) available resources. When the VSF with resource requirements of (10, 10, 10) is deployed, the resource utilization rate is $(1 + 1 + 0.5)/6 \cong 0.4167$ when placed in node A, and $(0.5+0.5+0.5)/6 = 0.25$ when deployed in node B. In short, when deploying a VSF server, it should be distributed as much as possible and at the same time prevent an increase in the utilization rate of a specific resource type within a specific node to obtain a low resource utilization rate.

The resource utilization results are similar except for the BFD in which VSFs are sorted in descending order based on resource requirements and then placed in the node where the remaining resources are expected to be the least. BFD can be seen as an algorithm that is not suitable for this multi-dimensional bin packing problem owing to its operating characteristics. Overall, the average utilization rate of the proposed algorithm is approximately 0.5 to 1% higher than that of FFD because it is effective in many situations to place a VSF server near the geographic center of the topology, and thus the resource utilization rate of the central nodes is relatively high. In addition, it is often necessary to deploy large servers that are less efficient because of the VSF reconfiguration. However, compared to the high traffic load reduction rate of more than 40%, the increase in the resource utilization rate can be considered an acceptable level.

Fig. 5(e) shows the success rate of the algorithm according to the number of services. As resource utilization increases, the success rate of Fit-type algorithms, which are primarily aimed at deployment, remains somewhat higher than the proposed algorithm. The success rate of w0 is the lowest among the variations of the proposed algorithm, which means that if the weights are not considered, a VSF that is rejected once is likely to be continuously rejected even if it is reconfigured with another VSF.

Next, to evaluate how the change in the number of clients affects the performance of each algorithm, a simulation was performed on the number of clients per service. Similar to the simulation of the number of services, because the maximum usage per client is fixed, an increase in the number of clients leads to an increase in the total usage. In the previous simulation, the maximum number of clients per service was five, and in this simulation, the number of services was 10. Therefore, it can be seen that the total amount of usage is approximately twice as much as that of the previous simulation when the same amount of change was applied.

Fig. 6(a) shows the traffic load according to the number of clients. Although the overall traffic load increased, it can be seen that it is linearly proportional, as in the previous simulation. Fig. 6(b) shows the traffic load reduction ratio of major algorithms compared to FFD according to the number of clients. As the number of clients increases, the total usage increases compared to the previous simulation, and hence, it can be seen that the traffic efficiency compared to FFD is further reduced. Fig. 6(c) shows the average traffic control efficiency versus FFD of major algorithms according to the number of clients. The SVSF showed a high level of traffic reduction at 29.96%, and the variations of the proposed algorithm showed a higher level of reduction at an average of 42.90%. It should be mentioned that the relatively high traffic reduction rate of SVSF is the distortion caused by the extremely low success rate. This is because only some cases in which deployment is successful due to low client usage parameters selected among 100,000 simulations are reflected in the average, and most cases in which the algorithm fails are

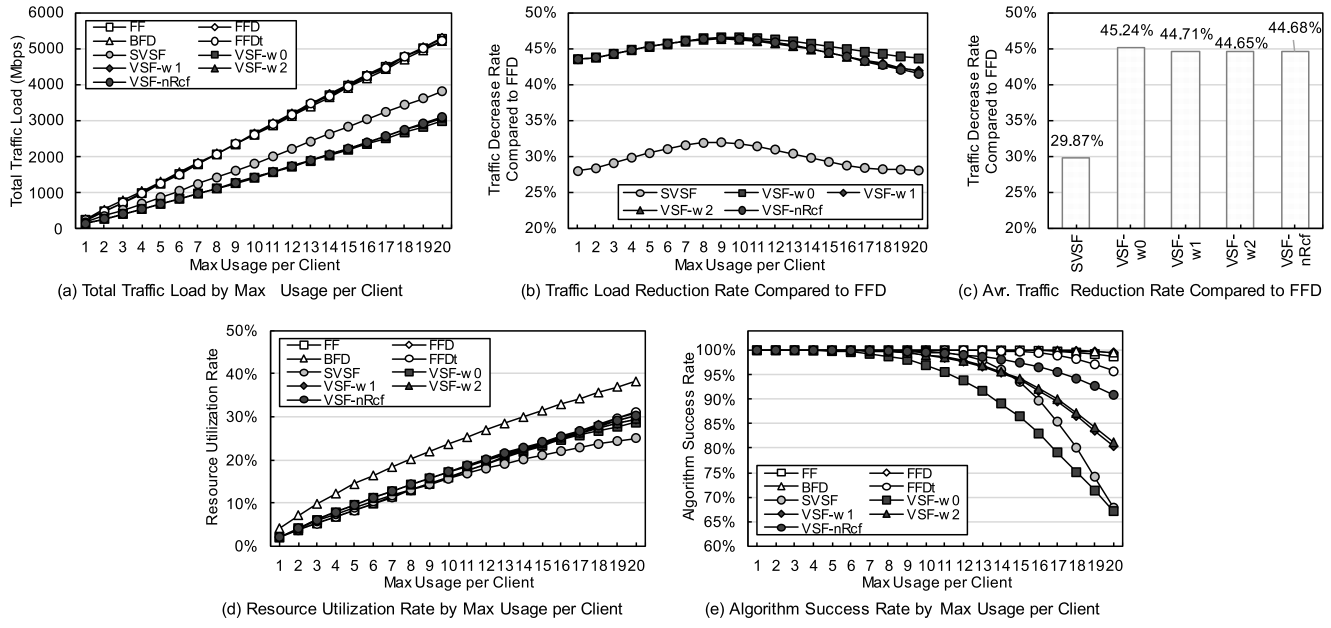


FIGURE 7. **Simulation results by the maximum usage level per client.**

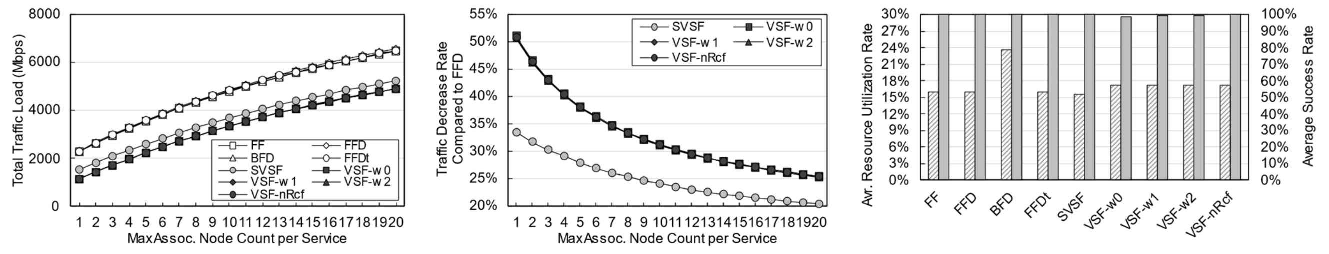
not reflected. In order to ignore distortion, simulation results are not displayed if the success rate is less than 50%.

Fig. 6(d) shows the resource utilization rate for each algorithm according to the number of clients. Except for SVSF, the results are similar to those of the previous simulation. The low usage rate of SVSF can be considered as distortion due to the low success rate as well. Fig. 6(e) shows the success rate for each algorithm according to the number of clients. SVSF shows a phenomenon in which the success rate rapidly decreases as the number of clients increases. This proves that when a single VSF is operated for each service, the VSF becomes lumpy, and thus it is impossible to flexibly cope with the resource constraints. Other algorithms based on multiple granular VSFs have a relatively high success rate. In the case of VSF variations, the success rate is lower in the order of w0, w1, w2, and nRcf. Compared to w1, w0 has a lower success rate because the weight factor is not considered, and w2 has a slightly higher success rate due to the heavy weight factor. The nRcf variation is a technique that only increases the weight without VSF reconfiguration, which means that the VSFs are kept fine-grained, thereby increasing flexibility. However, because VSFs with a large vector angle are more often placed, resource inefficiency is higher than other variations. In the case of this simulation, nRcf is 2.09% compared to w0, 1.56% compared to w1, and 1.15% higher than w2 for a section with a success rate of 80% or more.

In a situation where the number of services and the number of clients of service are maintained, the performance of the algorithms according to the change in the maximum usage of the client is examined. Fig. 7(a) shows the traffic load in the entire network according to the maximum usage per client. In the previous simulations where the maximum usage was 10, the number of VSFs increased according to the parameter change. However, in this simulation, when the maximum usage exceeds 10, the total usage of individual VSFs increases rather than the number of VSFs. Nevertheless, the results are similar to those of the previous simulations, as can be seen in Fig. 7(a), (b), and (c). In terms of traffic load, it was confirmed that SVSF decreased by 29.87% and the proposed algorithms by an average of 44.82% compared to the result of the FFD.

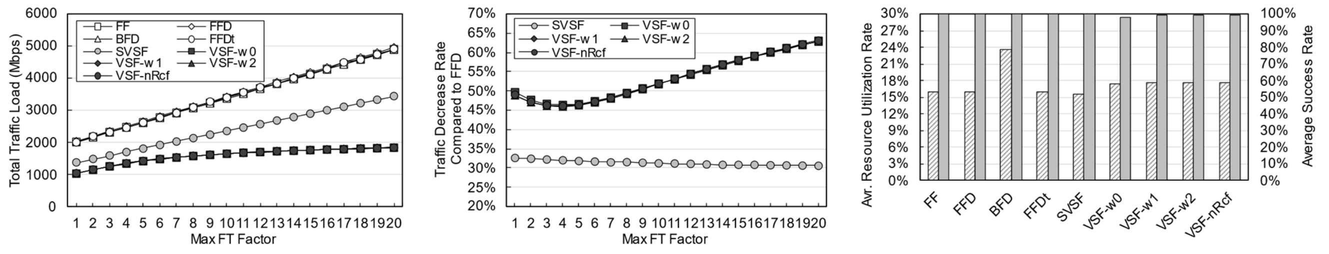
Fig. 7(d) shows the resource utilization rate by an algorithm according to the maximum usage by the client, and compared with the previous results, it can be seen that there is no significant difference. Fig. 7(e) shows the success rate for each algorithm. In the SVSF algorithm, which deploys only one VSF for each service, as the total usage of each VSF increases, the deployment success rate decreases because an appropriate server node cannot be found. Compared with Fig. 5(e), it can be seen that when the total usage in the network is at the same level, the success rate is higher when the number of VSFs increases than when the usage of VSF increases. Because the proposed algorithm generates fine-grained VSFs and then deploys them, it can be confirmed that the performance does not differ greatly, regardless of the situation in which the number of VSFs increases or the size of individual VSFs increases.

From Fig. 8, the simulations were performed while changing the maximum number of data sources for each service and the traffic load factor per usage under a constant total usage, where the total usage refers to the sum of usage generated by all clients. Unlike the traffic types FT and BTG, BTS is affected by both the total usage and the number of association



(a) Total Traffic Load by Maximum Assoc. Node Count (b) Traffic Load Reduction Rate Compared to FFD (c) Key Performance Results by Maximum Assoc. Node Count

FIGURE 8. **Simulation results showing the performance change of various algorithms by the maximum number of association nodes.**



(a) Total Traffic Load by Maximum FT Factor (b) Traffic Load Reduction Rate Compared to FFD (c) Key Performance Results by Maximum FT Factor

FIGURE 9. **Simulation results showing the various network performance indicators for each algorithm by the change in the maximum FT factor.**

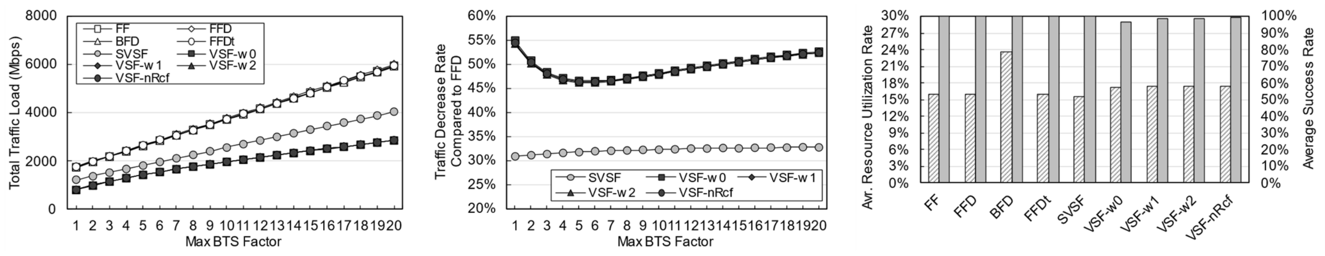
nodes. In Fig. 8(a), it can be seen that as the number of association nodes per service increases, the BTS increases and the total traffic load in the network also increases. This is because an increase in the number of association nodes means an increase in the average number of links used by individual VSFs. The total traffic load is the sum of the traffic load for each link. That is, an increase in the average number of links used by the VSF leads to an increase in the average traffic load for each link, resulting in an increase in the total traffic load.

Fig. 8(b) shows the traffic reduction rate of major algorithms compared to FFD. Unlike the previous simulations, as the simulation parameter increases, the traffic reduction rate compared to FFD decreases. In particular, it was confirmed that the difference in the traffic reduction rate was large compared to the simulation of the number of clients. When the total usage increases as the number of clients increases, the proposed algorithm can flexibly respond by configuring multiple VSFs. However, when the total usage increases because of the number of association nodes, only the average traffic usage of the VSF increases, and the flexibility in terms of the traffic control of the algorithm is reduced. In other words, if there are a large number of association nodes, it can be seen that the efficiency of the proposed algorithm that configures the VSF based on the client node decreases. This can be overcome by configuring hierarchies such as middleware nodes for collection purposes.

Fig. 8(c) shows the average resource utilization rate (left) and success rate (right) of each algorithm. It was confirmed that the resource utilization rate showed similar values, except for BFD. Because the total usage is constant, results similar to those obtained with specific parameter values in the previous simulation are shown. The success rate was close to 100% in most cases. This is also because the total usage is fixed, so the total computing resource capacity of the network is sufficient to handle the required resources of VSFs in all situations.

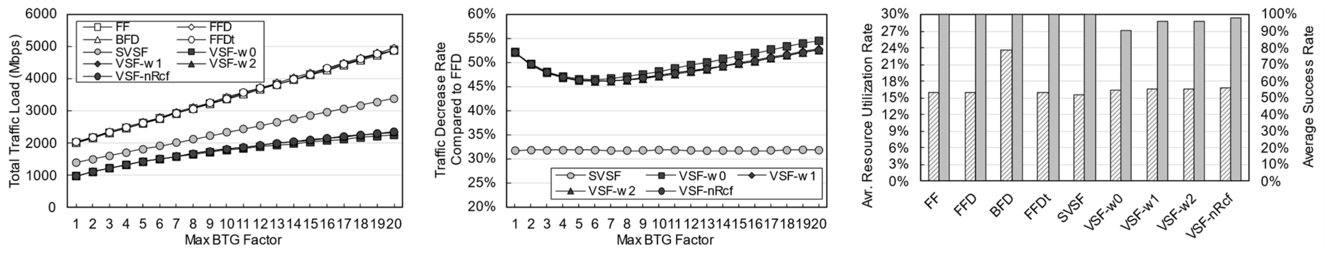
Fig. 9(a) shows the traffic load when the traffic factor of the FT changes. This reflects the situation where the client usage is the same as in the previous simulation, but the FT factor is increasing. As mentioned earlier, usage is a correlated relationship with traffic, but does not refer to the same concept. For example, usage can be the same concept as hours of use. Different FT factors for specific usage can be compared to a chat service and a video streaming service. As the FT factor increases, the resulting traffic load of FF, FFD, BFD, and FFDt increases significantly, and the traffic load of SVSF shows a lower level, nevertheless, it increases linearly. Thus, in the proposed algorithm, it can be seen that the increasing amount of traffic load gradually decreases as the FT factor increases. When the drafts and VSFs are configured in the proposed algorithm, the concept of the ETL, which consists of the sum of FT, BTS, and BTG according to the VSF server location is used. That is, a proposal to reduce FT by placing the server node closer to the client nodes becomes the best draft. Through this approach, the proposed algorithm can effectively cope with the increase in the FT factor.

Fig. 9(b) shows the traffic reduction rate of major algorithms compared to FFD. When the FT factor is low, BTS and BTG traffic are relatively important, and hence, the effect of FT is not significantly emphasized. However, if the maximum



(a) Total Traffic Load by Maximum BTS Factor (b) Traffic Load Reduction Rate Compared to FFD (c) Key Performance Results by Maximum BTS Factor

FIGURE 10. **Simulation results by the maximum BTS factor change.**



(a) Total Traffic Load by Maximum BTG Factor (b) Traffic Load Reduction Rate Compared to FFD (c) Key Performance Results by Maximum BTG Factor

FIGURE 11. **Simulation results by the maximum BTG factor change.**

FT factor is increased to 5 or more, it can be seen that the performance of the proposed algorithm is relatively higher than that of other simulation results. Fig. 9(c) shows the resource utilization and success rate for each algorithm. As a simulation in which only network traffic changes while maintaining the total usage, the results do not differ by parameter and by algorithms.

Fig. 10(a) shows the traffic load when the BTS factor changes. Like the previous simulation, as the BTS factor increases for the same usage, the resulting traffic load of FF, FFD, BFD, FFDt, and SVSF increases linearly, but the increasing amount of the variations gradually decreases as the factor changes. In the situation where the BTS factor is increasing, it can be seen that the overall traffic load is increased compared to the simulation result of increasing the FT factor. This is because the BTS affects more broadly as the VSF cannot be separated by the association node. When the VSF is initially configured, a specific VSF covers only one client node. In this case, if the FT is increasing, ETL can be greatly reduced by simply placing the server close to the client node. However, the association node is not a separate entity when configuring the VSF; therefore, if several association nodes are located at both extremes of the network topology, even if the location of the server node is adjusted, the ETL cannot be significantly reduced. For this reason, the overall traffic load increases compared to the previous simulation for the FT factor.

In Fig. 10(b), similar results appear in the traffic reduction rates of major algorithms compared to FFD. When the BTS factor is 1 or 2, the proposed algorithm shows higher performance compared to other values in this simulation, but as the factor increases, the performance improvement decreases and then increases again. This is certainly a high level, but it shows a relatively low efficiency compared to the simulation results related to the FT factor. That is, the proposed algorithm shows that the efficiency of the response to the increase in the factor of BTS that is not separated by VSF is relatively low rather than the response to the increase in the FT factor. Fig. 10(c) shows the resource utilization and success rate by the algorithm. Compared with Fig. 9(c), it can be observed that as the BTS factor increases, the effective VSF server placement of the proposed algorithm becomes hard due to the difficulty in responding to the BTS.

Fig. 11 shows the results of the simulation by changing the BTG factor. The result of the traffic load shown in Fig. 11(a) and the traffic reduction rate of major algorithms compared to the FFD in Fig. 11(b) show a similar trend to that of the BTS factor simulation. However, as the BTG factor increases, the difference in performance among the major algorithms appears distinct. Fig. 11(c) shows the resource utilization and success rate by the algorithm. Compared to the BTS factor simulation, the success rate of the proposed algorithms is somewhat lower, and the reason is as follows. The gateway node is given as a random node in each simulation, but as the BTG factor increases significantly, the priority of drafts for placing servers near the gateway node increases. Evidently, the number of VSFs that attempt to place the server on the gateway or nearby nodes increases, causing a computing resource bottleneck near the gateway. As a result, there are several VSFs that have drafts that are valid only for nodes

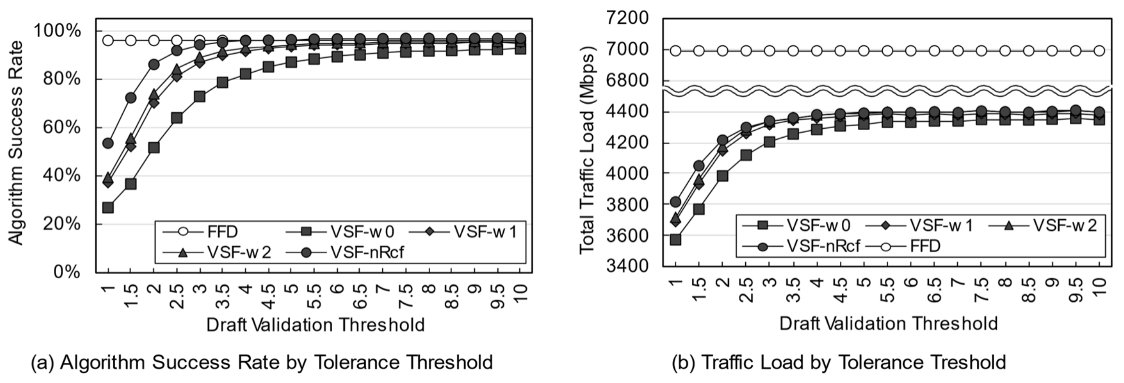


FIGURE 12. **Tradeoff between success rate and traffic load based on tolerance threshold when draft validation.**

near the gateway, and a few of them are more likely to be rejected, reducing the success rate.

Fig. 12 shows the traffic load and success rate of variations of the proposed algorithm according to the change in the tolerance threshold used for draft validation. For clearly checking the difference in success rate, the maximum number of clients per service was set to 15, and default parameter values were used for the rest. For comparison, the results of the FFD in the same situation are shown together. The average success rate of FFD is 96.08%, and the average total traffic load is 6,969.81 Mbps.

Fig. 12(a) indicates the change in the success rate according to the threshold change. It can be seen that the success rate of the algorithm increases as the number of drafts available for each VSF increases. From the threshold of 6 or more, it can be seen that all variations have a success rate of 90% or more, and it can be noted that certain variations exceed the success rate of FFD. Overall, the variation nRcf has the highest success rate and w0 has the lowest success rate. The success rate increases as the VSF is kept fine-grained and more weights are applied.

Fig. 12(b) indicates the total traffic load change according to the threshold change. The total traffic load increased notably until the threshold was increased from 1 to 6, but after that, the amount of change decreased significantly. This result shows that even if the number of drafts of all VSFs increases, only a few of them select drafts with high ETL. From another perspective, it may be a reason that there are not many drafts with an ETL exceeding six times the ETL of the ideal draft in a given topology. In a given simulation environment, even in the worst case, the traffic load can be reduced by 40% or more compared to FFD. Success rate and traffic load can be viewed as indicators that are highly coupled. This draft tolerance threshold value can serve as an optimization target when applied to an actual network.

Summarizing the simulation results, it was confirmed that the proposed algorithm can efficiently control traffic within the network regardless of the number of services or the number of clients. In particular, it was confirmed through FT simulation that it can cope with the service with high traffic load between the clients and the edge server very effectively. When the service usage level in the network increases rapidly and the available computing resources of each node become insufficient, the deployment success rate decreases. The proposed algorithm can reduce the overall traffic load in the network even at the expense of the success rate by adjusting the draft tolerance threshold to a low level. The success rate and traffic load may be optimized depending on the available computing resources in the network. Fortunately, the time complexity of the proposed algorithm is lower than that of the linear optimization algorithm. Therefore, if the algorithm fails in a specific situation, it is possible to complete the deployment within a reasonable time even if the operation is performed several times while changing the draft tolerance.

The vitalization of edge computing accelerates the utilization rate of edge services, and accordingly, related traffic within the edge network can also increase exponentially. When the edge network becomes congested, the low latency as the main advantage can be worsen. When the available network resources become insufficient, the probability of occurrence of phenomena that adversely affect transmission latency such as queuing delay, packet loss, and connection blocking increases. Therefore, the reduction of network congestion by the proposed scheme can complement the advantage of low latency of edge computing and can increase the overall reliability of the edge network.

V. CONCLUSION AND FUTURE WORK

Edge computing can provide network services with low latency and real-time processing by operating cloud services at the network edge. Edge computing has numerous advantages such as low latency, consideration of local characteristics, and localization of network congestion, however, owing to its inherent hierarchical, distributed, and heterogeneous nature, resource management related to service provision has become a significant challenge. Therefore, in this paper, an algorithm for deploying edge servers considering computing resources and traffic load is proposed. The proposed algorithm is based on the concept of VSF, which is a virtual network consisting of clients, data sources, server instance, and a gateway for a specific service. The VSF organizes drafts according to the expected server location, and they are sorted based on each estimated traffic load. Then, the edge server deployment of the VSFs is performed. Server deployment operates based on a weighted VBP technique in which the VSF coverage is considered, and an operation for selecting the next draft or reconfiguring for the rejected VSF is performed until the process is completed.

The proposed algorithm was compared by simulation in various aspects with our previous work and other schemes that determine server location in a specific order based on the required and available computing resources. It was confirmed that the proposed algorithm can significantly reduce the traffic load in the network regardless of the number of services or the number of clients. This means that under various circumstances of the network, the traffic load can be effectively reduced compared to existing deployment techniques. In particular, it was confirmed that it can respond very effectively to services with high traffic between the client and the edge server.

In the future, we plan to conduct research to further refine the system model. In addition to the concrete system model, the VSF configuration method based on this model will conduct research that can reflect the characteristics of the actual network content well. Through this, we intend to increase the reliability of simulation results for the metropolitan area network, which is close to the actual network usage environment, especially smart cities. Moreover, we will continue to contribute to the issue of integrated management of heterogeneous resources in edge computing by exploring a new algorithm that selects the advantages of the merge method and the segmentation method when configuring and reconfiguring the VSFs.

REFERENCES

[1] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[2] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–15.

[4] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 169–186.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[6] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[7] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, 2015, pp. 1–16, no. 11.

[8] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.

[9] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, Jul./Aug. 2013.

[10] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[11] T. El-Ganainy and M. Hefeeda, "Streaming virtual reality content," 2016, *arXiv:1612.08350*. [Online]. Available: http://arxiv.org/abs/1612.08350

[12] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, Mar. 2018.

[13] X. Ge, L. Pan, Q. Li, G. Mao, and S. Tu, "Multipath cooperative communications networks for augmented and virtual reality transmission," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2345–2358, Oct. 2017.

[14] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[15] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Netw.*, vol. 33, no. 4, pp. 162–169, Jul. 2019.

[16] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[17] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. M. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[18] L. Wang, L. Jiao, T. He, J. Li, and M. Muhlhauser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 468–476.

[19] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.

[20] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1910–1920, Aug. 2017.

[21] Y. Xu, V. Mahendran, and S. Radhakrishnan, "SDN docker: Enabling application auto-docking/undocking in edge switch," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 864–869.

[22] P. Bellavista and A. Zanni, "Feasibility of fog computing deployment based on docker containerization over RaspberryPi," in *Proc. 18th Int. Conf. Distrib. Comput. Netw.*, Jan. 2017, pp. 1–10.

[23] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–21, Mar. 2019.

[24] L. Shi, J. Furlong, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2013, pp. 9–15.

[25] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2647–2660, Nov. 2014.

[26] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[27] M. Mukherjee, V. Kumar, S. Kumar, R. Matamy, C. X. Mavromoustakis, Q. Zhang, M. Shojafar, and G. Mastorakis, "Computation offloading strategy in heterogeneous fog computing with energy and delay constraints," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–5.

[28] J. H. Lee, S. H. Chung, and W. S. Kim, "Fog server deployment technique: An approach based on computing resource usage," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 1, 2019, Art. no. 1550147718823994.

[29] R. Chaudhary, N. Kumar, and S. Zeadally, "Network service chaining in fog and cloud computing for the 5G environment: Data management and security challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 114–122, Nov. 2017.

[30] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Comput. Sci. Rev.*, vol. 24, pp. 63–79, May 2017.

[31] J. Csirik, "On the multidimensional vector bin packing," *Acta Cybernetica*, vol. 9, no. 4, pp. 361–369, 1990.
[32] D. S. Johnson and M. R. Garey, "A 7160 theorem for bin packing," *J. Complex.*, vol. 1, no. 1, pp. 65–106, Oct. 1985.
[33] A. Lewis, S. Ghosh, and N. Tzeng, "Run-time energy consumption estimation based on workload in server systems," in *Proc. USENIX HotPower*, Dec. 2008, pp. 1–5.
[34] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[35] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.
[36] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Annu. ACM Symp. Theory Comput. (STOC)*, 1984, pp. 302–311.
[37] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE Conf. Comput. Commun., 18th Annu. Joint Conf., IEEE Comput. Commun., Societies Future Now (INFOCOM)*, vol. 1, Mar. 1999, pp. 126–134.
[38] E. Chlebus and G. Divgi, "The Pareto or truncated Pareto distribution? Measurement-based modeling of session traffic for Wi-Fi wireless Internet access," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2007, pp. 3625–3630.

WON-SUK KIM (Member, IEEE) was born in Seoul, South Korea, in 1985. He received the B.E., M.S., and Ph.D. degrees in electrical and computer engineering from Pusan National University, Busan, South Korea, in 2010, 2012, and 2017, respectively.

From 2017 to 2019, he was the CEO of an AR content corporation called PlonVei, Busan. From 2019 to 2020, he was an Assistant Professor with the School of Digital Culture and Contents, Youngsan University, Busan. He is currently an Assistant Professor with the Department of Multimedia Engineering, Andong National University, Andong, South Korea. He has authored 27 articles and eight patents. His research interests include wireless mesh networks, software-defined networking, edge/fog computing, and augmented reality.

Dr. Kim was a recipient of the Minister's Award, the Highest Award in Creative Talent Evaluation in 2017.

SANG-HWA CHUNG (Member, IEEE) was born in Busan, South Korea, in 1960. He received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 1985, the M.S. degree in computer engineering from Iowa State University, Ames, IA, USA, in 1988, and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA, in 1993.

From 1993 to 1994, he was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA. Since 2016, he has been the Director of Dong-Nam Grand ICT Research Center. He is currently a Professor with the Computer Engineering Department, Pusan National University, Busan. He has authored over 240 articles and 70 patents. His research interests include embedded systems, wireless networks, software-defined networking, and smart factory.

Dr. Chung received the Best Paper Award from ETRI Journal in 2010 and the Engineering Paper Award of Pusan National University in 2011. In 2017 and 2019, he was selected as an Excellent Research Professor of the Computer Engineering with Pusan National University.

CHANG-WOO AHN (Member, IEEE) received the B.E., M.S., and Ph.D. degrees in electrical and computer engineering from Pusan National University, Busan, South Korea, in 2009, 2011, and 2019, respectively.

He is currently a Senior Engineer with Samsung Heavy Industries. His research interests include the Internet of Things, software-defined networking, and edge/fog computing.