Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(1)

DOI: 10.1177/1550147718823994
journals.sagepub.com/home/dsn
SAGE

Fog server deployment technique: An approach based on computing resource usage

Jung-Hoon Lee, Sang-Hwa Chung and Won-Suk Kim

Abstract

Cloud computing is a type of Internet-based computing that allows users to access computing resources that are connected to the Internet anytime and anywhere. Recently, as the Internet-of-Things market using the cloud has grown, a tremendous amount of data has been generated, and services requiring low latency are increasing. To solve these problems, a new architecture called fog computing has been proposed. Fog computing can process data on a network device close to the user, drastically reducing the bandwidth required from the network and providing near real-time response. However, not much research has been done on which network devices should be used to deploy the fog server. In this article, we propose a fog server deployment technique to minimize the data movement path in a fog computing environment and a technique to make full use of the computing resources of a fog device through a vector bin packing algorithm in a situation where many services are concentrated on one network device. Experimental results show that the proposed algorithm can reduce the data movement distance and maximize the utilization of the computing resources of the fog device.

Keywords

Fog computing, Internet of things, vector bin packing

Date received: 29 July 2018; accepted: 1 December 2018

Handling Editor: Francesco Longo

Introduction

Cloud computing[1] is a type of Internet-based computing that enables on-demand network access to computing resources (e.g., networks, servers, storage, and applications) anywhere (Figure 1). Users access the cloud through the Internet and receive the services they need. Users mainly perform input/output through personal devices, and the processing and storage of actual data is done in the cloud.

Recently, the Internet of things (IoT) market using the cloud has been growing.[2,3] IoT refers to a technology in which devices are connected to the Internet to exchange information between people and things, and between objects. The cloud is a good platform for IoT. As the IoT market grows, a lot of data will be generated by IoT devices. However, configuring and using a platform for analyzing and storing a large amount of data is inefficient because of the cost of constructing and maintaining the new platform. Therefore, it is effective to rent and use computing resources through a cloud service.

Despite these advantages, however, the combination of the cloud and IoT[4] can cause many problems. Indeed, by 2020, approximately 50 billion IoT devices will be connected to the Internet. However, if an

Pusan National University, Busan, Republic of Korea

Corresponding author:
Sang-Hwa Chung, Pusan National University, 6404, Engineering Bldg #6, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Republic of Korea.
Email: shchung@pusan.ac.kr

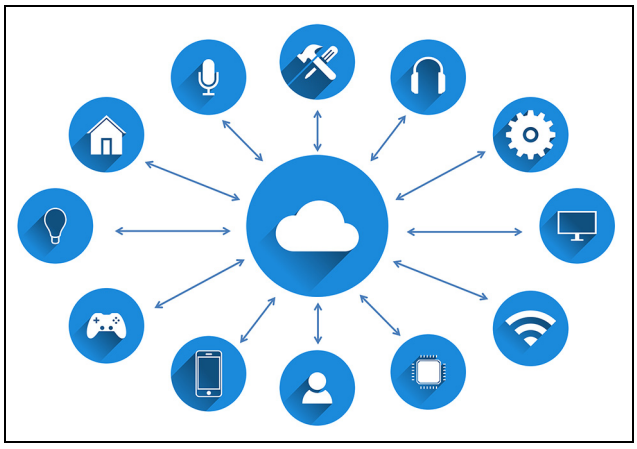


Figure 1. Cloud computing concept map.

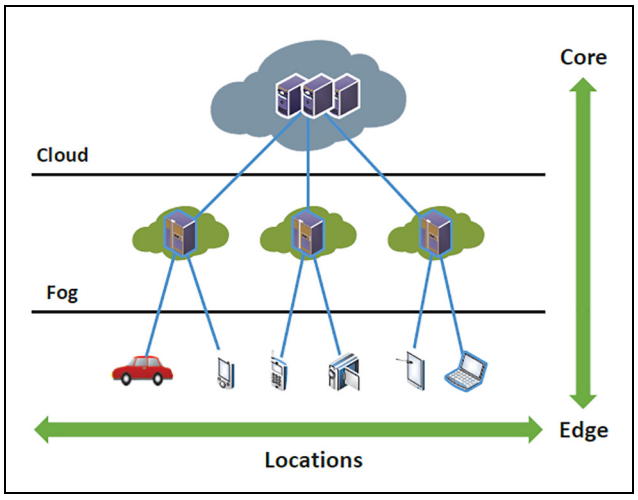


Figure 2. Fog computing layer.

exponentially growing number of IoT devices are connected to the Internet and send a lot of data to the cloud, there will be tremendous traffic across the network. The time and expense incurred in consuming a large amount of bandwidth in the cloud, and the time and expense required for storing and analyzing that data will also increase exponentially. Moreover, as various services are created, the number of services that require real-time processing will also increase. However, because the cloud is physically remote, there is a limit to the provision of these real-time services.

To overcome these problems, a new paradigm called fog computing has emerged.[5–7] Fog computing is a new computing model that has been proposed to enable massive traffic and real-time services resulting from a combination of IoT and cloud computing. The core concept of fog computing is that some services in the cloud are handled by network devices (routers, switches, hubs, and WiFi routers) that are present at the edge of the network, as shown in Figure 2. Creating a fog layer between the cloud and the endpoints can provide real-time services by processing service requests closer to where the data are generated, drastically reducing the amount of data delivered to the cloud. However, fog computing also presents some considerations.

First, fog computing handles services in network devices close to the IoT devices. However, network devices are relatively inefficient in terms of computing resources compared to the cloud, making it difficult to handle many services on a single network device. Therefore, fog computing is suitable for locally distributed, small-capacity processing, rather than centralized large-capacity processing. Hence, it is necessary to deploy the service in a proper location considering the computing resources of the network equipment.

Second, the service response time is drastically reduced when the service is processed in a fog computing environment rather than a service processing environment in the cloud. However, even in a fog computing environment, the service response time may be further reduced depending on where the service is processed. In addition, traffic on the network can be reduced by allowing services to be handled at a location close to the IoT devices. Research on fog computing architecture and fog computing platforms have been actively carried out, but there is a lack of research on where to deploy the service.

Third, it is necessary to consider the requirements of the various computing resources for each service. For example, a service that collects and pre-processes data from Internet devices has a relatively low requirement for computing resources. However, services that perform image processing or execute complex algorithms use more central processing units (CPUs) and memory because they have greater computational complexity than other services. In addition, the cache server and the database server require a relatively small amount of CPUs and memory, but a relatively large amount of storage space. When deploying these services, deploying only services that use a lot of resources on one network device can result in a lot of other resources being wasted.

Fourth, CPU utilization of cloud servers by day of week and hourly is quite variable, as shown in Figure 3. Because fog computing also handles cloud services on the network, continuing to service at fixed locations can lead to inefficient results. Therefore, it is necessary to relocate the service periodically according to the different service requests for each time zone.

In this article, we propose a practical service deployment method that can provide faster service response time, considering the various problems and utilize the computing resources of network equipment as efficiently

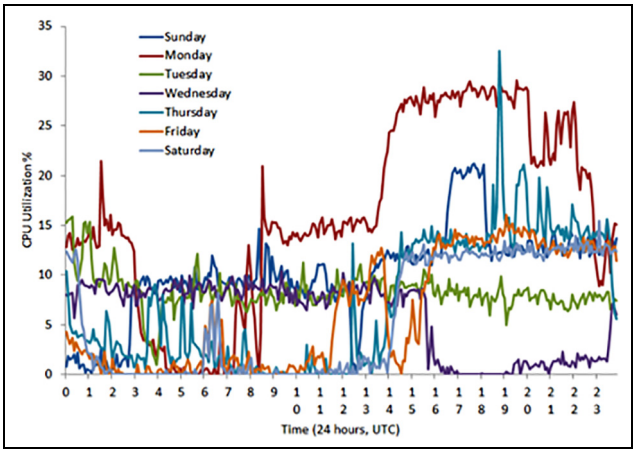


Figure 3. Host server CPU utilization of the Amazon E2C cloud.

as possible to process more services in a fog computing environment.

This technique has the following contributions:

First, by assigning the service to a location close to the user using the service, it can provide fast service response time and reduce network traffic.

Second, it efficiently utilizes the resources of the network device through a packing algorithm that makes it possible to use the computing resources of the network device as much as possible.

Third, it can periodically relocate according to the changing service environment for each time zone, thereby continuously providing fast service response time in a changing environment.

Related work

Research into deploying virtual machines in the cloud through vector bin packing algorithms has been active in the past. Lei et al.[8] has reduced power consumption in data centers by reducing the number of physical machines, in which virtual machines are deployed using vector bin packing algorithms. In addition, they compared various algorithms that could be used for packing and studied which algorithms were efficient. However, in order to consider the computing resources of fog devices that require relatively few computing resources in a fog computing environment besides CPU and memory, storage should also be considered. In addition to low computing resources compared with the cloud, for example, in the case of a cache server, storage requirements are high, so consideration of the three computing resources CPU, memory, and storage is necessary. Song et al.[9] studied the deployment of virtual machines in the cloud by extending the bin packing problem to multiple dimensions. They defined the variable item size bin packing (VISBP) algorithm and extended it to three dimensions to compare it with various algorithms.

However, there is a difference between deploying a virtual machine in such a cloud and deploying a fog server in a fog computing environment. In the cloud, the number of physical machines is fixed and kept at a minimum while virtual machines are implemented. However, because fog computing operates on a network device, it is important to deploy the services on the most efficient device, rather than reducing the power consumption because the network device must continue to operate even if the network device used is minimized. Therefore, in this article, we propose an algorithm that assigns services to the fog devices that are closest to users and enables as much use of fog device computing resources as possible through packing algorithms.

In the research related to fog computing, research on service implementation using fog computing architecture, fog device platforms for fog server deployment and execution, and fog application migration technique according to client patterns have been performed. However, there is a lack of research on which fog server should be deployed on which fog device.

Hu et al.[10] implemented a face recognition system in the fog computing environment. After processing the image received from the fog node adjacent to the user, the extracted face identifier was transmitted to the cloud. Distributing computation tasks from the cloud to the fog nodes improved overall processing efficiency and reduced network transmission. However, there was a fog node adjacent to the user, and there was no consideration as to which fog node should actually deploy the service.

Xu et al.[11] proposed a fog server deployment automation framework that analyzed packets for service requests and obtained the necessary applications from a repository. This study proposed a fog device platform that supports various services but does not consider computing resources and deploys a fog server for only a single user connected to the fog device.

Bellavista and Zanni[12] build real fog middleware using message queuing telemetry transport (MQTT) protocol in fog computing environment. It facilitates interoperability and portability of services because they are configured in the form of a Docker Container. In addition, it is capable of running multiple containers on nodes with limited resources such as RaspberryPi and has excellent scalability. However, this article proposes a fog computing platform for deploying services, but the study of service deployment location is left as a future study.

Mahmud et al.[13] propose a latency-aware application module management policy that targets both deadline-based quality of service (QoS) and applications and resource optimization. The proposed

management policy satisfies QoS of service application with deadline. In addition, it investigates how to optimize the number of resources without violating QoS of the applications. It is explained that if the application modules of a particular fog node are relocated to other fog nodes, that particular node can be turned off. However, in general, services in a fog computing environment operate on network devices, so even if there are no application modules to be executed, the devices do not turn off because they perform functions of the network device itself. In addition, if the application module is moved to reduce the number of operating fog nodes, the service delay time becomes longer. In this article, we propose a fog server deployment technique that allows all services to have low latency considering all fog nodes.

Fog server deployment technique

In this section, we propose a fog server deployment technique to determine where to deploy services. In section "Service flow," we introduce the concept of service flow for expressing a specific service. In section "Vector Bin Packing Model," we introduce the vector bin packing model for deploying services with computing resources in mind. In section "Fog server deployment technique," we describe the fog server deployment technique proposed in this article. The fog server deployment technique consists of two processes. First, we propose an algorithm for determining the fog server deployment location that allocates the service to the nearest distance from the devices constituting the service to provide fast service response time. Second, we propose a vector bin packing algorithm to efficiently deploy services considering computing resources when services are concentrated on a specific location.

We define two terms to take into account the location of the network device to process the service. First, a network device that processes a service is defined as a fog device. When a fog device is selected through the algorithm, the service actually operates on the network device, which is called a fog server. The IoT device receives or fetches the data from the data source, processes the service, and returns the result to the user.

Service flow

From this point on, fog server deployment is based on the concept of service flow. A service flow is formed between clients and data sources for processing any service that clients request. In addition, when a service is processed based on data collected from IoT devices, a service flow is formed between the IoT devices. As a result, a service flow is formed between devices associated with a service.

And the service flow further includes CPU, memory, and storage used to process the service. In the network topology of Figure 4, if there are service flows as shown in Table 1, f1 is one service flow between 5, 7, 10, 16, 17, and 19, and CPU, memory, and storage 15, 18, and 22 are used.

More specifically, in the case of the service flow f1, the clients 5 and 17 request any service and data for processing the service at 7, 16, and 19. And CPU, memory, and storage for processing the service are used as 15, 18, and 22.

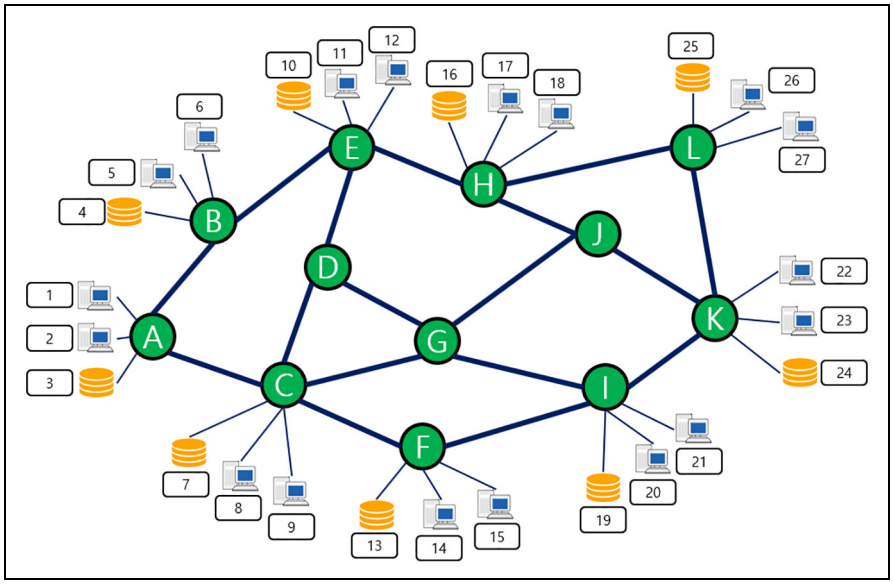


Figure 4. Network topology.

Table 1. Example service flows.

Service flow	Devices					CPU	Memory	Storage
f1	5	7	16	17	19	15	18	22
f2	3	7	8	25	27	13	13	25
f3	10	16	17	20	25	17	16	26

CPU: central processing unit.

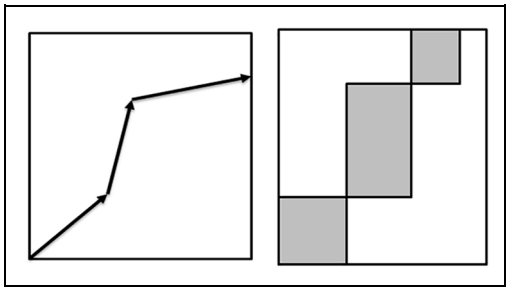


Figure 5. One-dimensional and two-dimensional vector bin packing.

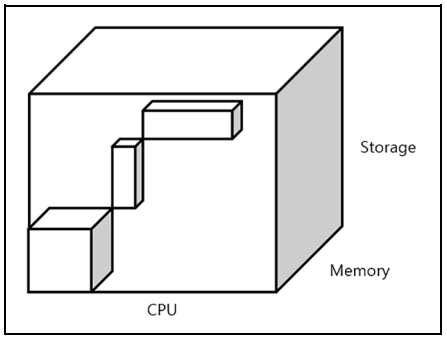


Figure 6. Three-dimensional vector bin packing.

Vector bin packing model

Vector bin packing is a widely used model for virtual machine deployment algorithms in the cloud. As shown in Figure 5, vector bin packing can pack services such that the sum of each computing resource does not exceed the total capacity. Thus, the vector bin packing model is suitable for deploying services with varying requirements to servers of known total capacity.

The vector bin packing model is extended to three dimensions as shown in Figure 6. The packing problem can be modeled by filling the fog servers corresponding to each service flow, considering the x-axis as the CPU, the y-axis as the memory, and the-z axis as the storage.

By filling the CPU, memory, and storage diagonally without overlapping, fog servers can be deployed without exceeding the total computing resources.

However, unlike a one-dimensional packing problem, we cannot use the popular first-fit decreasing (FFD) algorithm because we have to consider three aspects: CPU, memory, and storage. Therefore, we propose a vector bin packing algorithm to use a fog device's computing resources as efficiently as possible.

Fog server deployment technique

The algorithm for the fog server deployment technique is shown as Algorithm 1. The fog server deployment technique proposed in this article is divided into two processes: an algorithm for determining the fog server deployment location and a vector bin packing algorithm.

Algorithm for determining fog server deployment location. By line 18 of the algorithm, the fog server is assigned at the location where the sum of the distances from the devices corresponding to the service flow to the respective fog devices is the minimum, in order to determine the location of the fog server. A flowchart for the algorithm to determine the fog server deployment location is shown in Figure 7.

Through a simple example, the algorithm up to line 18 is described. When the service flows of Table 2 are in the topology of Figure 4, a method of determining the fog device on which to deploy the fog server is described. First, the shortest distances from the service flows to each fog device is calculated to determine the fog server deployment location. The shortest distance from each device $\{1, 2, 3, \ldots, 26, 27\}$ to each fog device in Figure 4 is shown in Table 3.

Table 3 shows the shortest distance from each device to each fog device by Dijkstra's algorithm. Based on Table 3, the sum of the shortest distances from the devices corresponding to each service flow to each fog device is calculated as shown in Table 4. Service flow f1 includes devices 16 and 24. Since the sum of the shortest distances from 16 and 24 to the fog device A is $4 + 5 = 9$, the distance from f1 to the fog device A is 9. In the same way, distanceTable (Table 4) is generated by calculating the distance from all service flows to the fog devices.

In Table 4, the shaded areas are fog devices that are closest in distance to each service flow. The service flows (distanceTable) are arranged in ascending order

Algorithm 1. Fog server deployment algorithm

```
1.  Input:
2.  Service_Flows = { f1, f2, …, fn }
3.  Fog_Devices = { d1, d2, …, dm }
4.  minDistance = ∞
    // Algorithm for determining fog server deployment location
5.  For each i ← { 1, …, n }
6.    For each j ← { 1, …, m }
7.      distanceTable[i][j] = getMinimumDistance(fi, dj)
8.    End For
9.  End For
10. sortServiceFlows (distanceTable)
11. For each i ← { 1, …, n }
12.   For each j ← { 1, …, m }
13.     If (distanceTable(i][j) < minDistance) then
14.         minDistance = distanceTable[i][j]
15.     End If
16.   End For
17.   assignServiceFlow(fi, d distance = = minDistance, nrAssigned = = minNrAssigned)
18. End For
    // Vector Bin Packing Algorithm
19. While (!All of ResourcesFogDevice<ResourcesServiceFlows)
20.   For each d ← Fog_Devices
21.     If (ResourcesdResourcesAssignedServiceFlows)
22.       For each f ← Assigned Service Flows
23.           angle = getAngleBetweenVectors(f, CurrentResource)
24.           If (angle < minAngle) then
25.               minAngle = angle
26.           End If
27.       End For
28.       For each f ← Assigned Service Flows
29.           If (getAngleBetweenVectors(f, CurrentResource) == minAngle) then
30.             deployFogServer(f)
31.           End If
32.       End For
33.     End If
34.     AssignNextFogDevice(Other_service_flows)
35.   End For
36. End While
37.
    getAngleBetweenVectors(f, c)
38.   dotProduct = f.cpu*c.cpu + f.mem*c.mem + f.storage*c.storage
39.   magnitudeF = sqrt(f.cpu*f.cpu + f.mem*f.mem + f.storage*f.storage)
40.   magnitudeC = sqrt(c.cpu*c.cpu + c.mem*c.mem + c.storage*c.storage)
41.   angle = cos-1(dotProduct/magnitudeF*magnitudeC)*180/PI
42.   return angle
43. End getAngleBetweenVectors
```

based on the number of fog devices closest in distance to determine the fog server deployment position. The number of fog devices closest to f6–f20 is 1, f7 and f11 are 2, … , and f2 is 5, so it is sorted in ascending order as shown in Table 5.

```
For each i ← { 1, …, n }
  For each j ← { 1, …, m }
    If ( distanceTable(i][j) < minDistance) then
        minDistance = distanceTable[i][j]
    End If
  End For
    assignServiceFlow(fi, d distance = = minDistance&&)
End For
```

When the order of the service flows is determined as shown in Table 5, the assignment starts from the service flow having the smallest number of fog devices that have the closest distance.

Because there is only one fog device closest in distance from f6 to f20, a fog server is assigned to the corresponding fog device. From f7, it is assigned to the fog device with the smallest service flow assigned in the previous step. Because fog device E has one service flow

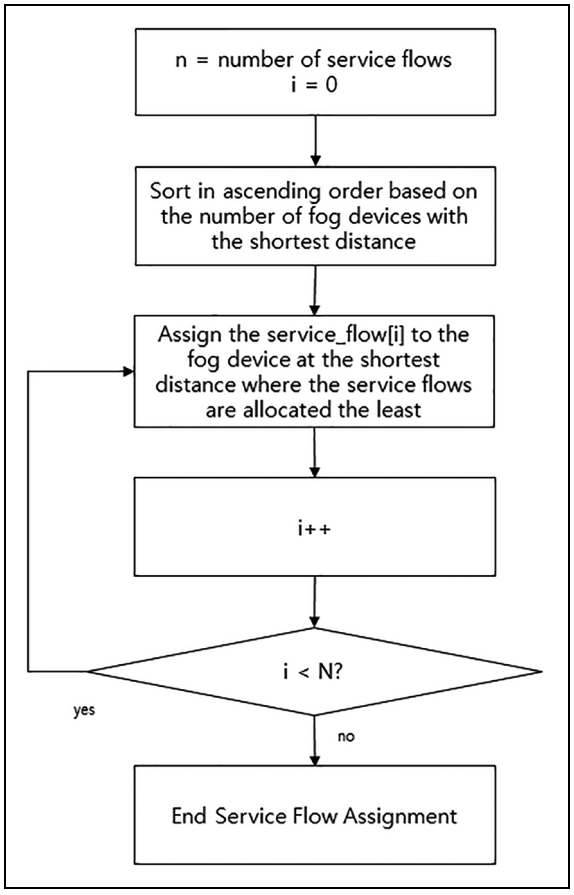


Figure 7. Flowchart of algorithm to determine fog server deployment location.

and H has three service flows, f7 is assigned to E. If the number of assigned service flows is the same, the fog server is assigned to the first fog device. After the fog server deployment location determination is completed in the following steps, it is finally assigned as shown in Tables 5 and 6.

The service response time and the network traffic can be reduced by assigning the fog server to the fog device closest to the service flow. In addition, a fog device with a few assigned service flows is selected so that the service flows are not concentrated on one fog device.

The algorithm for determining the fog server deployment location calculates the distances from each end device to each fog device through Dijkstra's algorithm and aligns the distanceTables from the service flows to each fog device in ascending order so that the time complexity is $O(n^2)$. Thereafter, because only a simple calculation is performed in order to assign the service flows, the time complexity is not greatly affected.

However, even if the fog server is assigned in this way, many service flows may be concentrated on such as in the fog device C. In this case, some service flows must be assigned to the next-closest fog device in order to prevent overload.

From the 19th line of the algorithm, we propose a vector bin packing algorithm that takes into account the direction of the computing resource vector in order to deploy the fog device's computing resources as efficiently as possible.

Table 2. 20 service flows.

Service flow	Devices							CPU	Memory	Storage
f1	16	24						25	10	11
f2	3	17						26	11	17
f3	1	11						28	13	15
f4	11	27						25	14	17
f5	21	26						23	15	16
f6	9	14	25					10	24	15
f7	5	18	21					15	29	20
f8	7	8	27					13	24	10
f9	3	9	23					14	22	17
f10	6	9	19					19	26	17
f11	5	7	16	17	19			15	18	22
f12	5	16	18	20	21			12	14	28
f13	3	7	8	25	27			13	13	25
f14	5	13	16	21	26			19	15	24
f15	10	16	17	20	25			17	16	26
f16	3	4	5	12	17	18	24	28	22	19
f17	1	6	7	9	15	20	25	27	23	22
f18	8	12	17	78	23	25	26	24	21	28
f19	1	3	4	5	8	12	16	28	27	22
f20	6	7	9	12	14	22	26	26	26	30

CPU: central processing unit.

Table 3 The shortest distance from each device to the fog devices.

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	2	3	3	3	3	4	4	4	5	5
2	1	2	2	3	3	3	3	4	4	4	5	5
3	1	2	2	3	3	3	3	4	4	4	5	5
4	2	1	3	3	2	4	4	3	5	4	5	4
5	2	1	3	3	2	4	4	3	5	4	5	4
6	2	1	3	3	2	4	4	3	5	4	5	4
7	2	3	1	2	3	2	2	4	3	3	4	5
8	2	3	1	2	3	2	2	4	3	3	4	5
9	2	3	1	2	3	2	2	4	3	3	4	5
10	3	2	3	2	1	4	3	2	4	3	4	3
11	3	2	3	2	1	4	3	2	4	3	4	3
12	3	2	3	2	1	4	3	2	4	3	4	3
13	3	4	2	3	4	1	3	5	2	4	3	4
14	3	4	2	3	4	1	3	5	2	4	3	4
15	3	4	2	3	4	1	3	5	2	4	3	4
16	4	3	4	3	2	5	3	1	4	2	3	2
17	4	3	4	3	2	5	3	1	4	2	3	2
18	4	3	4	3	2	5	3	1	4	2	3	2
19	4	5	3	3	4	2	2	4	1	3	2	3
20	4	5	3	3	4	2	2	4	1	3	2	3
21	4	5	3	3	4	2	2	4	1	3	2	3
22	5	5	4	4	4	3	3	3	2	2	1	2
23	5	5	4	4	4	3	3	3	2	2	1	2
24	5	5	4	4	4	3	3	3	2	2	1	2
25	5	4	5	4	3	4	4	2	3	3	2	1
26	5	4	5	4	3	4	4	2	3	3	2	1
27	5	4	5	4	3	4	4	2	3	3	2	1

distanceTable[i][j] = getMinimumDistance(f_i, d_j).

Table 4. Shortest distances from service flows to each fog device.

	A	B	C	D	E	F	G	H	I	J	K	L
f1	9	8	8	7	6	8	6	4	6	4	4	4
f2	6	6	7	7	6	7	7	6	7	7	7	6
f3	4	4	5	5	4	7	6	6	8	7	9	8
f4	8	6	8	6	4	8	7	4	7	6	6	4
f5	9	9	8	7	7	6	6	6	4	6	4	4
f6	10	11	8	9	10	7	9	11	8	10	9	10
f7	10	9	10	9	8	11	9	8	10	9	10	9
f8	9	10	7	8	9	8	8	10	9	9	10	11
f9	8	10	7	9	10	8	8	11	9	9	10	12
f10	8	9	7	8	9	8	8	11	9	10	11	12
f11	16	15	15	14	13	18	14	13	17	14	17	16
f12	18	17	17	15	14	18	14	13	15	14	15	14
f13	15	16	14	15	15	15	15	16	16	16	17	17
f14	18	17	17	16	15	16	16	15	15	16	15	14
f15	20	17	19	15	12	20	15	10	16	13	14	11
f16	21	17	23	21	16	28	23	17	28	21	26	22
f17	19	22	17	20	22	18	20	26	21	24	25	27
f18	28	24	26	22	18	27	22	15	23	18	19	16
f19	15	14	18	19	16	25	22	21	29	24	31	28
f20	22	22	19	20	20	20	21	23	22	22	23	24

sortServiceFlows (distanceTable).

Shades represented fog devices that are closest in distance from the service flow.

Table 5. Service flows that have been aligned and assigned.

	A	B	C	D	E	F	G	H	I	J	K	L
f6	10	11	8	9	10	7	9	11	8	10	9	10
f8	9	10	7	8	9	8	8	10	9	9	10	11
f9	8	10	7	9	10	8	8	11	9	9	10	12
f10	8	9	7	8	9	8	8	11	9	10	11	12
f12	18	17	17	15	14	18	14	13	15	14	15	14
f13	15	16	14	15	15	15	15	16	16	16	17	17
f14	18	17	17	16	15	16	16	15	15	16	15	14
f15	20	17	19	15	12	20	15	10	16	13	14	11
f16	21	17	23	21	16	28	23	17	28	21	26	22
f17	19	22	17	20	22	18	20	26	21	24	25	27
f18	28	24	26	22	18	27	22	15	23	18	19	16
f19	15	14	18	19	16	25	22	21	29	24	31	28
f20	22	22	19	20	20	20	21	23	22	22	23	24
f7	10	9	10	9	8	11	9	8	10	9	10	9
f11	16	15	15	14	13	18	14	13	17	14	17	16
f3	4	4	5	5	4	7	6	6	8	7	9	8
f4	8	6	8	6	4	8	7	4	7	6	6	4
f5	9	9	8	7	7	6	6	6	4	6	4	4
f1	9	8	8	7	6	8	6	4	6	4	4	4
f2	6	6	7	7	6	7	7	6	7	7	7	6

Shades represented fog devices that are closest in distance from the service flow.
Slashes represents the number of fog devices that are closest from the service flow.

Table 6. Location of service flows.

Fog devices	Service flow	Devices							CPU	Memory	Storage
A	f3	1	11						28	13	15
	f2	3	17						26	11	17
B	f19	1	3	4	5	8	12	16	28	27	22s
C	f8	7	8	27					13	24	10
	f9	3	9	23					14	22	17
	f10	6	9	19					19	26	17
	f13	3	7	8	25	27			13	13	25
	f17	1	6	7	9	15	20	25	27	23	22
	f20	6	7	9	12	14	22	26	26	26	30
E	f16	3	4	5	12	17	18	24	28	22	19
	f7	5	18	21					15	29	20
	f11	5	7	16	17	19			15	18	22
F	f6	9	14	25					10	24	15
H	f12	5	16	18	20	21			12	14	28
	f15	10	16	17	20	25			17	16	26
	f18	8	12	17	78	23	25	26	24	21	28
I	f5	21	26						23	15	16
J	f1	16	24						25	10	11
L	f14	5	13	16	21	26			19	15	24
	f4	11	27						25	14	17

CPU: central processing unit.

Vector bin packing algorithm. The flowchart for the vector packing algorithm is shown in Figure 8. The core of the algorithm deploys a service flow that is similar to the direction of the current computing resource vector. In the three-dimensional (3D) space, CPU, memory, and storage are considered as a one 3D vector, and the service flow closest to the direction of the computing resource vector remaining in the fog device is selected and deployed.

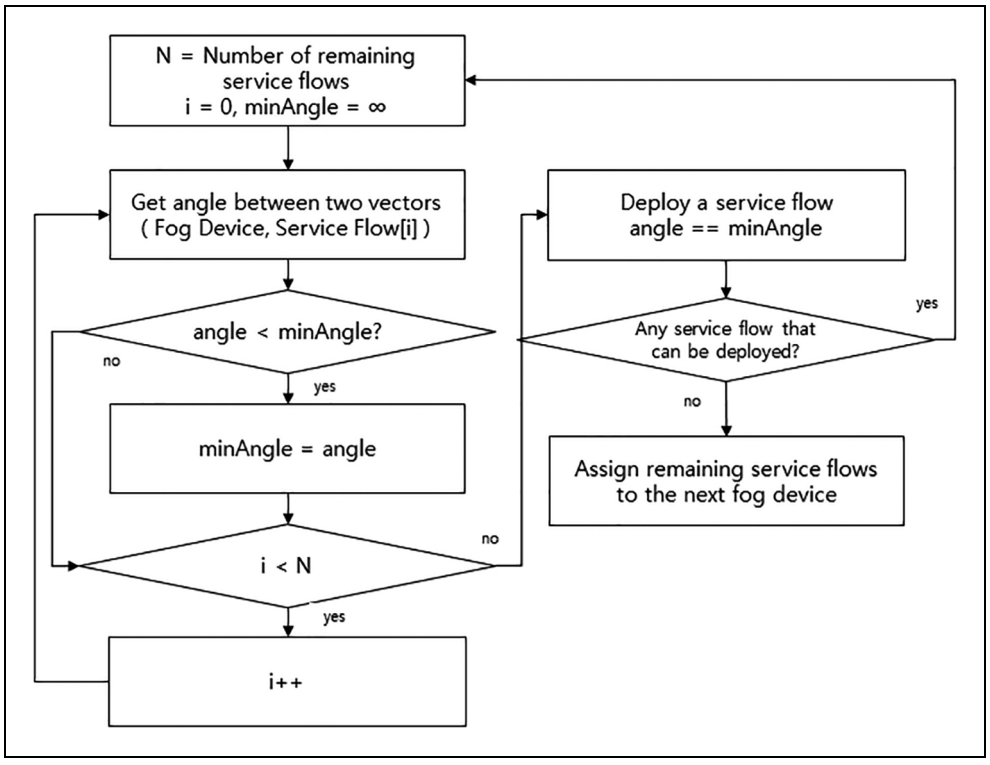


Figure 8. Flowchart of vector bin packing algorithm.

```
For each f ← Assigned Service Flows
  angle = getAngleBetweenVectors(f, CurrentResource)
  If ( angle < minAngle) then
      minAngle = angle
  End If
End For
For each f ← Assigned Service Flows
  If ( getAngleBetweenVectors (
    f, CurrentResource) = = minAngle) then
    deployFogServer(f)
  End If
End For
```

First, the angle of the computing resource vector of the remaining fog device and the angles of the assigned service flow vectors are calculated, and the service flow having the smallest angular difference is deployed in the fog device. Once a service flow is deployed, the same algorithm is repeated when there is a service flow that can be deployed next in the remaining computing resources.

If service flows are deployed and there are no more computing resources available, then the service flows will be assigned to the next nearest fog device (Table 5). If the sum of the computing resources of the assigned service flows is higher than the computing resources of the fog device, the above operation continues. After the service flows are assigned so as not to exceed the computing resources of all the fog devices, the fog server deployment is finally complete. Then, the software-defined networking (SDN) controller modifies the flow tables of the network devices so that the devices corresponding to each service flow and the fog server can communicate with each other.

In Table 6, all the assigned service flows can be deployed in all fog devices except for fog device C. However, since the sum of the computing resources required by the service flows assigned to the fog device C exceeds the resources (100, 100, 100) of the fog device, vector bin packing is performed in the fog device C.

Since the total computing resources of the fog device C are (100, 100, 100), the service flow f20 (26, 26, 30) with the closest angles are deployed. Thereafter, f17 (27, 23, 22) with the closest angle to the computing resources (74, 74, 70) of the remaining fog devices is deployed. Similarly, f9 and f10 are deployed.

After the four service flows are deployed, the remaining f8 and f13 are assigned to the next nearest fog device. Therefore, f8 is assigned to fog device D, and f13 is assigned to fog device A (see Table 5). The algorithm is then terminated because fog device D and A can accommodate both already deployed or newly assigned service flows.

Table 7. Computing resources by service flow—random.

Service	CPU	Memory	Storage
f1	47	14	17
f2	15	49	20
f3	48	47	42
f4	12	14	48
f5	1	15	42
f6	13	13	45
f7	48	10	11
f8	47	43	42
f9	23	29	4
f10	10	44	15
f11	41	35	32
f12	19	15	44
f13	48	42	49
f14	16	30	12
f15	3	2	18
f16	14	42	17
f17	44	41	48
f18	22	13	26
f19	17	16	46
f20	45	13	15
f21	42	11	17
f22	19	46	17
f23	15	18	36
f24	23	49	17
f25	46	46	40
f26	35	24	28
f27	48	11	19
f28	13	44	10
f29	12	39	2
f30	50	45	23
f31	32	3	50
f32	46	40	25
f33	43	15	16
f34	50	42	15
f35	15	18	42
f36	46	33	21
f37	2	22	46
f38	40	39	8
f39	42	36	1
f40	25	19	12

CPU: central processing unit.

If the newly assigned fog device cannot accept service flows, the vector bin packing algorithm is performed once again and repeats the same operation.

The vector bin packing algorithm has $O(n^2)$ time complexity because n, n–1, n–1, … , 1 angle calculations occur when n service flows are assigned to one fog device.

To see how many service flows can fit into a few fog devices via the vector bin packing algorithm, we compare it with the first-fit (FF) algorithm, which deploys service flows on the next fog device when no resources are available. There are 40 service flows as shown in Table 7. CPU, memory, and storage were randomly generated from 1 to 50.

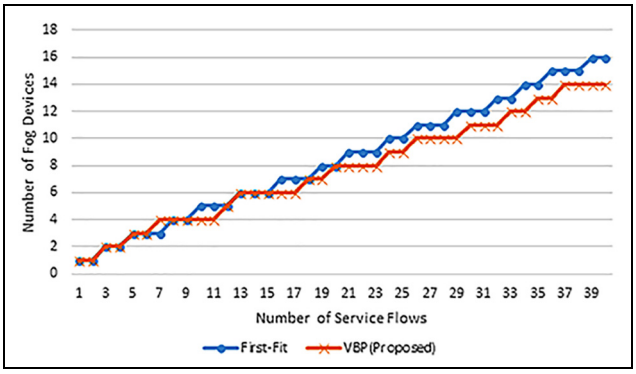


Figure 9. Number of fog devices required based on number of service flows.

Figure 9 shows the number of fog devices needed when 1 to 40 service flows, as shown in Table 7, are deployed. The capacity of the CPU, memory, and storage of the fog devices was assumed to be 100. With the FF algorithm, 16 fog devices were needed to accommodate 40 service flows, because if a fog device has no resources, then the service flow will be deployed on the next device. However, with the proposed vector bin packing algorithm, only 14 fog devices were needed to accommodate the entire service flow.

In particular, the vector bin packing algorithm is more effective when there are many service flows that use more specific resources. Table 8 shows 13 CPU-intensive service flows, 13 memory-intensive service flows, and 14 storage-intensive service flows.

Figure 10 shows the number of fog devices needed when 1 to 40 service flows, as shown in Table 8, are deployed. The number of fog devices required is reduced from 20 to 14 compared to the FF algorithm because when CPU-intensive service flows are deployed, the remaining space accommodates memory and storage-intensive service flows.

In practice, there are many services that require relatively specific resources rather than uniformly using all resources. The proposed algorithm makes it possible to efficiently use the computing resources of all the fog devices considering the resource requirement of the services in the fog computing environment.

In this manner, the minimum number of fog devices is used for each service by assigning it to a location that can provide the minimum response time. The vector bin packing algorithm packs such that the computing resources of the assigned service flows does not exceed the total computing resources of the fog device, so as much of the computing resources of the fog device as possible are used. By assigning unpacked service flows to the next-closest fog device and packing it, it is

Table 8. Computing resources by service flow—using more specific resources.

Service	CPU	Memory	Storage
f1	48	10	11
f2	42	11	17
f3	45	13	15
f4	47	14	17
f5	43	15	16
f6	48	12	19
f7	47	13	12
f8	44	11	18
f9	48	17	12
f10	46	16	10
f11	43	19	17
f12	48	11	19
f13	46	10	12
f14	10	44	15
f15	15	49	20
f16	13	44	10
f17	14	42	17
f18	19	46	17
f19	15	48	16
f20	12	43	16
f21	15	44	18
f22	11	45	12
f23	12	49	2
f24	10	45	13
f25	16	40	15
f26	12	43	10
f27	15	18	42
f28	12	14	48
f29	13	13	45
f30	19	15	44
f31	17	16	46
f32	10	12	45
f33	16	13	41
f34	1	15	42
f35	2	12	46
f36	3	2	48
f37	10	19	48
f38	13	19	44
f39	12	16	41
f40	15	19	42

CPU: central processing unit.

possible to assign as many service flows as possible to the nearest fog device, to obtain the following effects.

First, the response time of services can be reduced by assigning as many service flows as possible to the nearest fog device.

Second, the route to service processing is also reduced, so traffic on the entire network can be reduced.

Third, many service flows can be processed in the fog computing environment by packing the service flows to use as much of the computing resources of the fog devices as possible.

Fourth, by reassigning the fog server periodically according to the service environment by changing time zone, it is possible to continuously provide quick service response time in a changing environment.

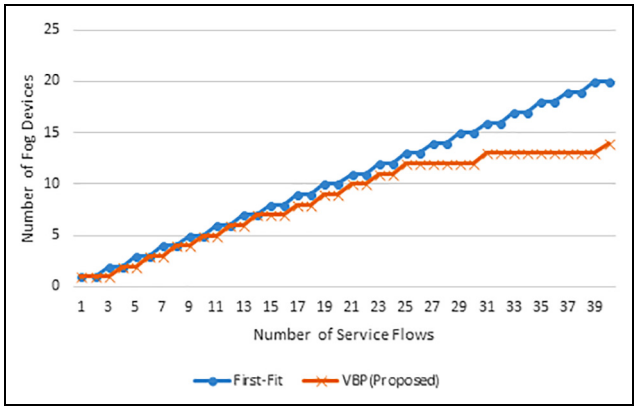


Figure 10. Number of fog devices required based on number of service flows.

Experimental results

Experiment environment

In section "Experimental results," we evaluated the performance of the fog server deployment technique described in section "Fog server deployment technique" through simulation. The network environment used in the experiment is shown in Figure 11. There are 12 fog devices from A to L to consider in the fog computing environment. In addition, assuming a general situation, there are two clients and one data source adjacent to the outside fog device.

Evaluation of fog server deployment technique

In order to evaluate the fog server deployment technique, we compare the results of using two algorithms for each case where 30 service flows are assigned to random fog devices and specific fog devices through an algorithm for determining the fog server deployment location.

The first algorithm is a sequential algorithm. If there are enough computing resources left to deploy a fog server in the fog device, assign a fog server, or assign it to the next fog device if there are not enough resources left. The second algorithm is the vector bin packing algorithm proposed in the article.

The computing resource capacity of all fog devices is assumed to be 100 for CPU, memory, and storage, and the computing resources required to process the service flow are set to random values between 1 and 50 for CPU, memory, and storage.

Random service flow environment. First, the service flow environment of Table 9 is used to evaluate the case where 30 service flows are assigned to random fog devices. In the service flow environment shown in Table 9, each service flow is assigned to a fog device through

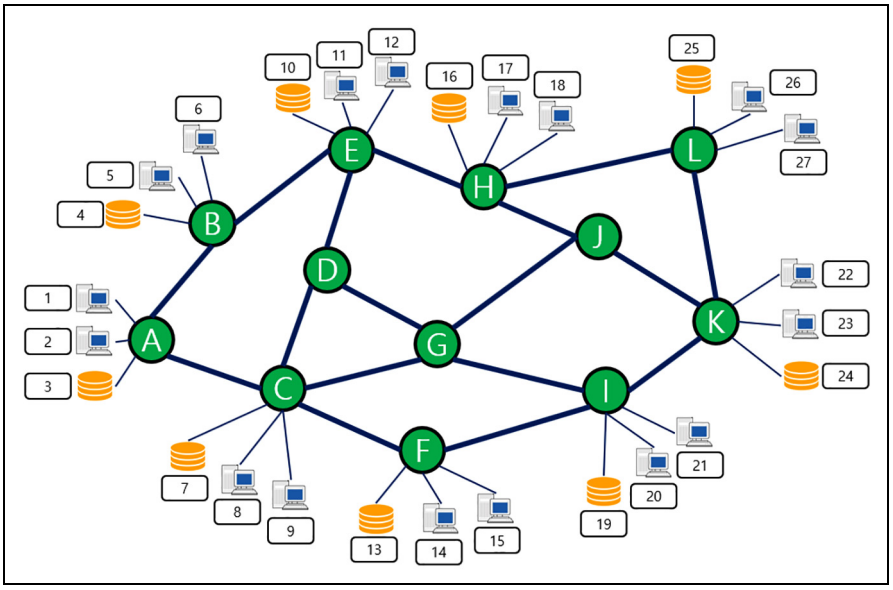


Figure 11. Network topology.

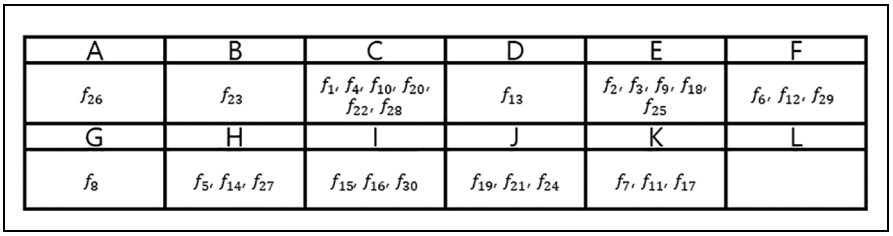
Table 9. 30 random service flow environments.

Service flow	Devices							CPU	Memory	Storage
f1	5	8	10	13	14	17	24	23	13	4
f2	2	9	12	13	17	26	27	33	32	46
f3	3	8	10	11	14	18	26	20	31	18
f4	6	7	9	12	13	14	17	18	6	14
f5	1	3	10	16	21	23	27	32	20	17
f6	1	8	13	14	18	22	25	40	48	7
f7	7	9	18	21	23	24	26	39	30	50
f8	2	7	9	17	19	22	23	42	7	29
f9	2	6	8	12	19	23	25	27	28	31
f10	1	3	7	9	11	13	26	38	43	35
f11	9	17	21	22	23	25	26	48	2	45
f12	4	6	8	14	20	22	24	6	49	10
f13	8	9	10	15	16	18	19	20	34	28
f14	1	2	5	16	22	23	25	11	45	25
f15	6	7	13	19	21	23	26	16	3	21
f16	2	10	14	18	20	21	24	44	3	39
f17	1	8	12	13	23	24	26	46	32	23
f18	1	4	5	12	16	21	22	6	38	41
f19	6	9	12	13	16	22	24	5	50	17
f20	3	4	7	8	14	19	25	22	36	12
f21	2	7	8	16	17	22	24	21	40	10
f22	7	8	9	14	15	18	26	6	44	28
f23	2	4	5	7	10	13	25	28	30	11
f24	9	11	12	17	22	23	24	5	10	35
f25	6	10	11	15	16	17	20	47	4	8
f26	2	3	5	6	7	11	14	24	29	16
f27	2	4	11	17	23	24	26	4	33	41
f28	1	6	7	9	22	26	27	23	20	9
f29	2	5	9	15	19	23	27	26	15	13
f30	7	10	14	15	18	19	20	6	3	7

CPU: central processing unit.

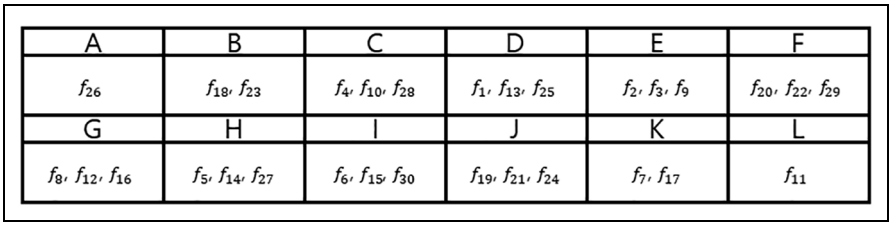
an algorithm for determining fog server deployment location and is assigned as shown in Figure 12.

For devices C, E, F, and K, the sum of the computing resource requirements of the assigned service flows



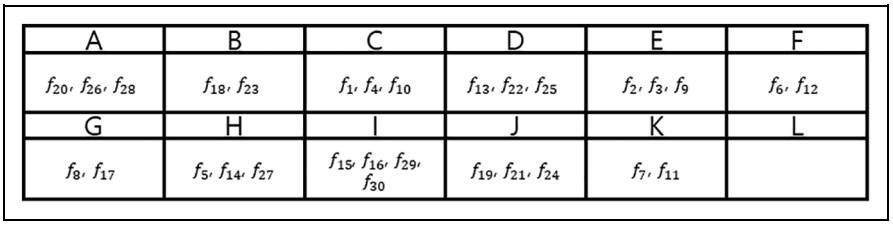
A	B	C	D	E	F
f_{26}	f_{23}	f_1, f_4, f_{10}, f_{20}, f_{22}, f_{28}	f_{13}	f_2, f_3, f_9, f_{18}, f_{25}	f_6, f_{12}, f_{29}
G	H	I	J	K	L
f_8	f_5, f_{14}, f_{27}	f_{15}, f_{16}, f_{30}	f_{19}, f_{21}, f_{24}	f_7, f_{11}, f_{17}	

Figure 12. The service flows assigned through the algorithm for determining the fog server deployment location.



A	B	C	D	E	F
f_{26}	f_{18}, f_{23}	f_4, f_{10}, f_{28}	f_1, f_{13}, f_{25}	f_2, f_3, f_9	f_{20}, f_{22}, f_{29}
G	H	I	J	K	L
f_8, f_{12}, f_{16}	f_5, f_{14}, f_{27}	f_6, f_{15}, f_{30}	f_{19}, f_{21}, f_{24}	f_7, f_{17}	f_{11}

Figure 13. Service flows deployed through vector bin packing algorithm.



A	B	C	D	E	F
f_{20}, f_{26}, f_{28}	f_{18}, f_{23}	f_1, f_4, f_{10}	f_{13}, f_{22}, f_{25}	f_2, f_3, f_9	f_6, f_{12}
G	H	I	J	K	L
f_8, f_{17}	f_5, f_{14}, f_{27}	f_{15}, f_{16}, f_{29}, f_{30}	f_{19}, f_{21}, f_{24}	f_7, f_{11}	

Figure 14. Service flows deployed through a sequential algorithm.

exceeds the fog device capacity, so the service flows are packed through the vector bin packing algorithm. Unassigned service flows are assigned to the nearest fog device at the next distance. The above process is repeated until the sum of computing resources of assigned service flows does not exceed the resources of all the fog devices. The completed deployment process is shown in Figure 13.

Figure 14 shows the final deployment location when deployed through a sequential algorithm. The sum of the distances from the entire service flows to the fog devices when deployed through the vector bin packing algorithm and sequential algorithm are both 548. Because the service flows are assigned equally in a random service flow environment, the sum of the distances is similar because the service flows can be arranged in a higher priority fog device.

Table 10 shows the distance from the service flows in Table 9 to the fog devices. The shaded area is the location where the service flows are finally deployed. The sum of the distances from all the service flows to the fog devices can go up to 772 in the worst case. However, since the algorithm for determining fog server deployment location first assigns the service flows at the shortest distance, the distance is shorter than when randomly deployed. Even after the vector bin packing algorithm is performed, a faster response time can be obtained because service flows are deployed in fog devices which are relatively close to the devices forming the service flow.

Figure 15 shows the distribution of computing resources when deployed through the vector bin packing algorithm, and Figure 16 shows the distribution of computing resources when deployed sequentially. For a more numerical representation, the standard deviations for three computing resources of all fog devices are shown in Table 11.

The environment in which service flows are assigned to specific fog devices. In the situation where the service flows are randomly generated, there is not much packing because each service flow is initially assigned uniformly. There is not a large difference in the standard deviation between the sequential deployment method and the method using the vector bin packing algorithm. However, because the packing attempts to use the three resources

Table 10. Shortest distances from service flows to each fog device.

	A	B	C	D	E	F	G	H	I	J	K	L
f1	22	22	19	20	20	20	21	23	22	22	23	24
f2	23	22	22	21	19	23	22	20	23	22	23	21
f3	21	20	20	19	17	23	21	20	24	22	25	23
f4	19	20	16	18	19	19	20	24	23	23	26	27
f5	23	23	23	22	20	24	21	20	22	21	22	21
f6	23	25	20	22	23	19	21	24	20	22	21	23
f7	27	28	22	22	23	21	19	21	18	18	17	20
f8	23	26	19	21	23	20	18	23	19	19	20	24
f9	22	22	21	21	20	22	21	22	22	22	23	23
f10	17	20	16	19	20	19	20	25	23	24	27	28
f11	30	29	26	24	23	23	21	19	18	18	15	16
f12	23	24	20	22	23	19	21	25	20	22	21	24
f13	22	23	18	18	19	21	18	21	21	20	23	24
f14	23	22	24	24	21	25	23	20	24	21	22	21
f15	25	27	21	22	24	18	20	25	17	22	19	22
f16	24	26	21	21	22	20	19	23	18	21	20	22
f17	24	25	21	22	22	20	21	23	20	21	20	22
f18	21	19	22	21	18	25	22	20	25	22	25	23
f19	24	23	21	21	20	22	21	21	22	20	21	22
f20	19	22	17	20	22	18	20	26	21	24	25	27
f21	23	24	20	21	21	23	19	20	22	18	21	23
f22	21	24	16	19	22	17	19	25	20	22	23	26
f23	18	17	19	20	18	22	23	23	26	25	28	26
f24	27	25	23	21	19	24	20	18	21	17	18	19
f25	23	20	22	19	16	25	21	18	24	21	24	21
f26	14	15	16	19	18	21	22	25	27	26	31	30
f27	25	22	25	23	19	26	23	18	24	20	21	19
f28	22	22	21	22	21	22	22	22	23	22	23	23
f29	22	24	20	22	23	19	21	25	20	23	22	24
f30	23	26	18	19	22	17	18	25	17	22	21	24

Service flows are finally deployed in the shaded location.

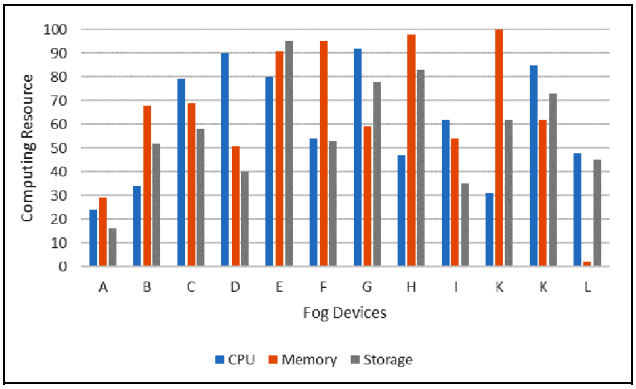


Figure 15. Distribution of computing resources when deployed through a vector bin packing algorithm.

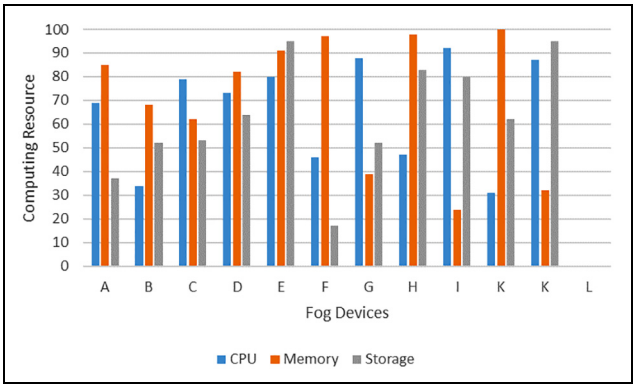


Figure 16. Distribution of computing resources when deployed through a sequential algorithm.

as efficiently as possible, the standard deviation is small when using the vector bin packing algorithm.

In the next experiment, fog servers are deployed in a situation where service flows are assigned to four fog

Table 11. Computing resource standard deviations for the results deployed through the two algorithms.

Fog device	A	B	C	D	E	F	Average
VBP (proposed)	5.35	13.89	8.58	21.45	6.34	19.57	
Sequential	19.96	13.89	10.78	7.35	6.34	33.07	
Fog device	G	H	I	J	K	L	
VBP (proposed)	13.52	21.4	11.32	28.22	9.39	21.01	15
Sequential	20.73	21.4	29.63	28.22	28	0	18.28

VBP: vector bin packing.

A	B	C	D	E	F
				f_6, f_8, f_{14}, f_{15}, f_{18}, f_{23}, f_{26}, f_{29}	f_3, f_5, f_{11}, f_{16}, f_{19}, f_{21}, f_{25}, f_{27}
G	H	I	J	K	L
		f_1, f_2, f_{10}, f_{12}, f_{22}, f_{24}, f_{30}		f_4, f_7, f_9, f_{13}, f_{17}, f_{20}, f_{28}	

Figure 17. The service flows assigned through the algorithm for determining the fog server deployment location.

A	B	C	D	E	F
f_5, f_{21}, f_{25}	f_8	f_1, f_{16}, f_{19}	f_{22}, f_{29}	f_6, f_{14}, f_{15}, f_{18}	f_{11}, f_{12}, f_{27}
G	H	I	J	K	L
f_3, f_9, f_{24}, f_{30}	f_{23}, f_{26}	f_2, f_{10}	f_{17}, f_{28}	f_7, f_{13}, f_{20}	f_4

Figure 18. Service flows deployed through the vector bin packing algorithm.

A	B	C	D	E	F
f_{21}, f_{25}	f_{10}, f_{18}	f_{16}, f_{19}, f_{27}	f_{22}, f_{29}	f_6, f_8, f_{14}	f_3, f_5
G	H	I	J	K	L
f_9, f_{12}, f_{30}	f_{15}, f_{23}	f_1, f_2	f_{13}, f_{17}, f_{28}	f_4, f_7, f_{24}	f_{20}, f_{26}

Figure 19. Service flows deployed through a sequential algorithm.

devices through an algorithm for determining fog server deployment locations. The computing resource usage of the service flows was randomly set to a value between 1 and 50.

When assigned through an algorithm for determining fog server deployment location, service flows were assigned to four fog devices as shown in Figure 17. Then, when the fog servers were deployed using the vector bin packing algorithm and the sequential method, the service flows were finally deployed as shown in Figures 18 and 19.

The sum of the shortest distances from all service flows to each fog device after deployment was 546 when using vector bin packing and sequential deployment. The reason why the sum of the shortest distances was the same even when the service flows were initially assigned to the four fog devices is because service flow 11 is not deployed in a fog

Table 12. Computing resource standard deviations for the results deployed through the two algorithms.

Fog device	A	B	C	D	E	F	Average
VBP (proposed)	13.14	10.66	3.56	26.51	6.55	1.7	
Sequential	26.84	6.48	6.18	26.51	24.1	31.59	
Fog device	G	H	I	J	K	L	
VBP (proposed)	0.82	29.33	6.13	12.39	8.34	17.02	11.34
Sequential	23.15	24.78	16.52	19.48	11.22	11.78	19.05

VBP: vector bin packing.

A	B	C	D	E	F
		f_{11}, f_{12}, f_{25}	f_5, f_8, f_{14}, f_{24}	f_{17}, f_{21}, f_{28}	f_{10}, f_{18}, f_{20}
G	H	I	J	K	L
f_{15}, f_{22}, f_{26}	f_4, f_6	f_2, f_7	f_{19}, f_{23}, f_{29}	f_{13}, f_{16}, f_{27}, f_{30}	f_1, f_3, f_9

Figure 20. Service flows deployed through the vector bin packing algorithm.

A	B	C	D	E	F
f_{26}, f_{27}	f_{28}, f_{29}, f_{30}	f_9, f_{10}, f_{11}	f_{16}, f_{17}, f_{18}	f_{21}, f_{22}, f_{23}	f_3, f_4
G	H	I	J	K	L
f_5, f_6	f_{24}, f_{25}	f_1, f_2	f_{12}, f_{13}, f_{14}, f_{15}	f_7, f_8	f_{19}, f_{20}

Figure 21. Service flows deployed through the sequential algorithm.

device during sequential deployment. Because of the inefficient use of computing resources, not all service flows can be deployed. Conversely, the vector bin packing algorithm packs the service flows to use the computing resources of the fog devices as efficiently as possible, so that all 30 service flows can be deployed and more service flows can be accommodated.

Table 12 shows the standard deviation of the computing resources for both deployment methods. Using the proposed vector bin packing algorithm, the standard deviation is as small as 7.71. The difference is greater than in the previous experimental random flow environment greater because there are more packings.

An environment in which service flows are assigned to one fog device. Finally, we conducted experiments in environments where the same devices were using 30 services. The service flows were initially assigned to fog device I by an algorithm for determining the fog server deployment location.

When 30 service flows are assigned to fog device I, the deployment is completed by two algorithms, as shown in Figures 20 and 21.

When the same devices used 30 services, all the service flows were assigned to one fog device through an algorithm that determines the fog server deployment location, and the remaining service flows after packing were all assigned to the nearest fog device. Therefore, the effect of the vector bin packing algorithm proposed in this article was the greatest.

When sequential deployment was used, service flows were deployed on all fog devices, but fog servers were not deployed on either A or B fog devices because of the efficient use of the fog device's computing resources when using the vector bin packing algorithm. Therefore, other service flows were more acceptable. The sum of the shortest distances from all service flows to each fog device after deployment was 611 using the vector bin packing algorithm, and 643 when deployed sequentially. Moreover, as shown in Table 13, the standard deviation for the computing resources of each fog device also decreased by an average of 8.65.

Table 13. Computing resource standard deviations for the results deployed through the two algorithms.

Fog device	A	B	C	D	E	F	Average
VBP (proposed)	0	0	7.12	7.04	6.55	8.04	
Sequential	5.25	6.38	10.68	22.07	7.85	33.18	
Fog device	G	H	I	J	K	L	
VBP (proposed)	6.34	12.36	4.5	8.64	11.58	4.64	6.4
Sequential	16.86	25.42	16.52	13.1	11.81	11.44	15.05

VBP: vector bin packing.

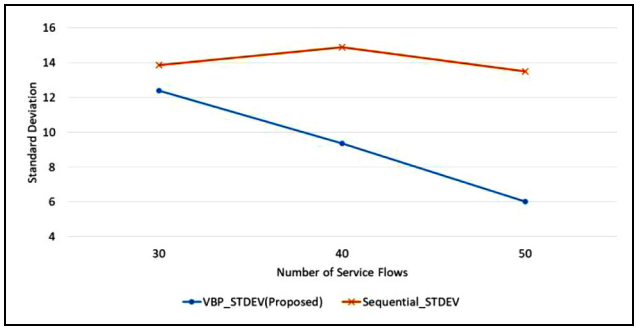


Figure 22. Standard deviation of computing resources after fog server deployment.

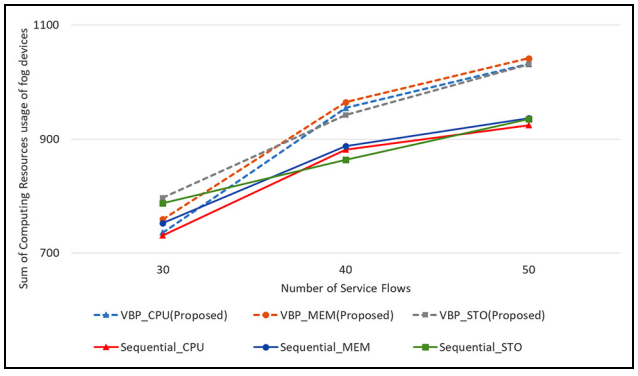


Figure 23. Sum of computing resource usage of fog devices after deploying fog server.

Experiment in general environment. However, in general, many users use various services, so the service flow is not concentrated in a specific fog device. Therefore, we conducted experiments in arbitrary service flow environments to take into account more general situations.

Experiments were conducted 100 times for each case in a situation where the number of service flows existing in the network environment was different. The standard deviation and usage of computing resources of fog devices were measured. The devices corresponding to the service flow were randomly generated from 1 to 27 (Figure 11), and the computing resources were randomly generated between 1 and 50.

Figure 22 shows the average of the standard deviation of the computing resource usage of fog devices after deploying the fog server. The standard deviation is lower when the vector bin packing algorithm (proposed in the article) is used rather than sequential deployment because the computing resources of the fog devices are used uniformly.

Figure 23 shows the sum of usage of computing resources of fog devices after fog server deployment. When the number of service flows existing in the network is small, the usage of computing resources is the same because all the service flows are deployed in the fog devices.

However, the larger the number of service flows, the more computing resources are consumed when using vector bin packing algorithms. This is because more service flows can be deployed in the network by considering the remaining amount of computing resources, so that the computing resources of the whole fog device is used as efficiently as possible.

Conclusion and future work

Conclusion

In this article, we proposed an algorithm for determining fog server deployment location and the vector bin packing algorithm to deploy various services in a fog computing environment. In addition, when clients, data sources, and IoT devices request a service or provide the data necessary for processing the service, the devices and the computing resources (CPU, memory, and storage) required to process the service were defined as one service flow. Based on the service flows, the service flows were assigned to the closest fog device to process the service. When the sum of the computing resources of the assigned service flows exceeded the total computing resources of the fog device, the service flows were packed through the vector bin packing algorithm to utilize as much of the computing resources of the fog device as possible. Unpacked service flows were assigned to the nearest fog device and packed again.

Thus, efficient use of computing resources makes it possible to operate as many services as possible in a fog computing environment. In addition, by reducing the distance from each service flow to the fog devices as much as possible, faster response time can be obtained, and the traffic on the network can also be reduced because the distance over which the data travel is shortened.

Experiments showed that the sum of the distances from service flows to fog devices was reduced, and the use of computing resources was much more efficient. We also confirmed that the proposed vector bin packing algorithm was more efficient, as many service flows were assigned to fog devices and many instances of packing occurred. Therefore, the algorithm proposed in this article efficiently packs many service flows into a fog computing environment so that many service flows can operate.

Future work

In this study, we implemented an offline algorithm for the current service flow environment, and periodically monitored the service flow environment to relocate the fog server. However, considering the change in the service flow environment in real time (the fog server is deleted after the service is terminated, and a new service is deployed in the remaining space), implementing an online algorithm would make it possible to more efficiently deploy the fog server. In addition, communication with the cloud is one of the factors to consider in fog server deployment, such as sending data from a fog computing environment to the cloud or receiving data from the cloud to process the service.

Future work will focus on communication with the cloud, and research on fog server deployment and migration techniques that are tailored to real-time service flow environments using online algorithms.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (grant no.: NRF-2018R1D1A1B07049355). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2018R1D1A1B07049355). This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2018-2016-0-00318) supervised by the IITP (Institute for Information & communications Technology Promotion).

ORCID iD

Jung-Hoon Lee https://orcid.org/0000-0002-7622-6546

References

1. Mell P and Grance T. The NIST definition of cloud computing, 2011.
2. Xia F, Yang LT, Wang L, et al. Internet of things. *Int J Commun Syst* 2012; 25(9): 1101–1102.
3. Atzori L, Iera A and Morabito G. The Internet of things: a survey. *Comp Netw* 2010; 54(15): 2787–2805.
4. Botta A, de Donato W, Persico V, et al. On the integration of cloud computing and Internet of things. In: *2014 international conference on future Internet of things and cloud (FiCloud)*, Barcelona, 27–29 August 2014. New York: IEEE.
5. Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the Internet of things. In: *Proceedings of the first edition of the MCC workshop on mobile cloud computing*, Helsinki, 17 August 2012. New York: ACM.
6. Bonomi F, Milito R, Natarajan P, et al. Fog computing: a platform for Internet of things and analytics. In: F Bonomi, R Milito, P Natarajan, et al. (eds) *Big data and Internet of things: a roadmap for smart environments*. Cham: Springer, 2014, pp.169–186.
7. Stojmenovic I and Wen S. The fog computing paradigm: scenarios and security issues. In: *2014 federated conference on computer science and information systems (FedCSIS)*, Warsaw, 7–10 September 2014. New York: IEEE.
8. Lei S, Furlong J and Wang R. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. In: *2013 IEEE symposium on computers and communications (ISCC)*, Split, 7–10 July 2013. New York: IEEE.
9. Song W, Xiao Z, Chen Q, et al. Adaptive resource provisioning for the cloud using online bin packing. *IEEE T Comput* 2014; 63(11): 2647–2660.
10. Hu P, Ning H, Qiu T, et al. Fog computing based face identification and resolution scheme in Internet of things. *IEEE T Ind Inform* 2017; 13(4): 1910–1920.
11. Xu Y, Mahendran V and Radhakrishnan S. SDN docker: enabling application auto-docking/undocking in edge switch. In: *2016 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, San Francisco, CA, 10–14 April 2016. New York: IEEE.
12. Bellavista P and Zanni A. Feasibility of fog computing deployment based on Docker containerization over RaspberryPi. In: *Proceedings of the 18th international conference on distributed computing and networking*, Hyderabad, India, 5–7 January 2017. New York: ACM.
13. Mahmud R, Ramamohanarao K and Buyya R. Latency-aware application module management for fog computing environments. *ACM Trans Internet Technol* 2018; 19: 9.