Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 6123234, 15 pages
http://dx.doi.org/10.1155/2016/6123234

Research Article

An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing

Aymen Abdullah Alsaffar,[1] Hung Phuoc Pham,[1] Choong-Seon Hong,[1] Eui-Nam Huh,[1] and Mohammad Aazam[2]

[1]*Department of Computer Engineering, Kyung Hee University, Yongin-si, Seoul, Republic of Korea*
[2]*Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada*

Correspondence should be addressed to Eui-Nam Huh; johnhuh@khu.ac.kr

Received 29 April 2016; Revised 25 July 2016; Accepted 25 August 2016

Academic Editor: Young-June Choi

Despite the wide utilization of cloud computing (e.g., services, applications, and resources), some of the services, applications, and smart devices are not able to fully benefit from this attractive cloud computing paradigm due to the following issues: (1) smart devices might be lacking in their capacity (e.g., processing, memory, storage, battery, and resource allocation), (2) they might be lacking in their network resources, and (3) the high network latency to centralized server in cloud might not be efficient for delay-sensitive application, services, and resource allocations requests. Fog computing is promising paradigm that can extend cloud resources to edge of network, solving the abovementioned issue. As a result, in this work, we propose an architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing. We provide new algorithm that is decision rules of linearized decision tree based on three conditions (services size, completion time, and VMs capacity) for managing and delegating user request in order to balance workload. Moreover, we propose algorithm to allocate resources to meet service level agreement (SLA) and quality of services (QoS) as well as optimizing big data distribution in fog and cloud computing. Our simulation result shows that our proposed approach can efficiently balance workload, improve resource allocation efficiently, optimize big data distribution, and show better performance than other existing methods.

1. Introduction

Cloud computing is not only a technology that continuously advances for offering a variety of services and resources to many cloud consumers smart devices (e.g., IoT, smart wearable devices, smart phone, smart tablets, and smart home appliances) but also an enabling developer to develop more applications, tools, and services. Cloud computing architecture can empower ubiquitous, advantageous, and on-demand network access to a shared pool of configurable computing resources, providing many other benefits (e.g., storages, services, applications, networks, virtualized resources, large scale computation, schedulable virtual servers, high expansibility, computing power, low price services, virtual network, network bandwidth, and high reliability) [1–3]. One of the technologies that is gaining popularity is known as Internet of things (IoT). IoT is a technology that is still developing and enables many objects (e.g., thin-client, smart phone, smart tablets, smart home appliances, smart wearable devices, and sensor) to connect to Internet to perform variety of services (e.g., memory, storage space, processing, virtualization, resource allocation, services delegation, surfing, send/receive big data, and viewing social sites). Thus, smart devices services are present in every aspect of our daily life (e.g., health care, medicine treatment, education, and remotely controlled smart devices). Cloud computing technology is being widely used to support variety of cloud consumer devices, services, and applications.

Despite the wide utilization of cloud computing (e.g., services, applications, and resources), some of the services,

applications, and smart devices are not able to fully benefit from this attractive cloud computing paradigm due to the following issues: (1) smart devices are lacking in their capacity (e.g., processing, memory, storage, battery, and resource allocation), (2) they are lacking in their network resources, and (3) the high network latency to centralized server in cloud is not efficient for delay-sensitive application, services, and resource allocations requests. According to [3], the number of devices that connected to Internet will exceed 24 billion by 2020. The rapid increase of number of Internet connected devices combined with the long distance between user smart devices and cloud computing, and the repeatedly requested services, will pose heavy burden to network performance and network bandwidth which in return will degrade cloud computing QoS as well. Moreover, the high network latency between user devices and cloud may not be ideal for delay-sensitive applications, services, and resources.

To resolve the abovementioned issues, we utilize fog computing which is a new paradigm that extends cloud computing resources and service to the edge of network. It is highly virtualized infrastructure that can provide networking services, computation, storage, memory between IoT devices, and traditional cloud computing environment [4]. Furthermore, fog computing is located in localized environment, making it closer to user location and giving it the advantage over cloud to provide variety of distributed applications [4]. The impressive advantages that fog computing offers over cloud computing will not only increase the number of requests services (i.e., delay-sensitive services, applications, and data) but also direct most of user requested services if not all to fog computing only. This will lead to unbalanced workload of services and degraded performance of fog performance, user requests, and the abandonment of IoT service from cloud computing.

Therefore, in this paper, we introduce new architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing. We provide new algorithm that is decision rules of linearized decision tree based on three conditions (services size, completion time, and VMs capacity) for managing and delegating user request. Furthermore, we present our new strategy for data distribution optimization such as big data. Moreover, we present an algorithm to perform resource allocation in order to satisfy service level agreement (SLA) and quality of service (QoS). Our simulation shows that our approach can improve the efficiency of resource allocation and show better performance comparing with other approaches.

The rest of paper is organized as follows. In Section 2, we introduce related work. In Section 3, we introduce our system architecture and motivating scenario. In Section 4, we present our proposed mechanism for service delegation, resource allocation, and big data distribution processes. In Section 5, we present our implementation and analysis result. In Section 6, we present our conclusion and future work.

2. Related Work

There are many researches attempting to resolve the abovementioned issues. In [5], the author introduces efficient synchronization in cloud for number of hierarchy distributed file systems. The author deploys the conception of master-slave architecture to propagate data to reduce traffic. In [6], the author introduces method for resource scheduling which can be efficient in mitigating the impacts that influence application respond time and utilization of the system. In [7, 8], the authors introduce the impact of data transmission delay on the performance. In [9], the author introduces one method to make a parallel processing for big data which can increase the performance in federated cloud computing. However, these researches do not mention how much resources should be utilized.

Also, there are many completed researches that deal with resources allocation. In [10], the authors explain through their work that shared allocation is superior to dedicated allocation. Nevertheless, the authors do not perform experiment with arbitrary number of SLAs. Moreover, authors do not show how fast the server needs to be in order to guarantee quality of service (QoS). In [11, 12], the authors provide services to large number of SLAs as it is quite difficult to obtain performance between shared and reserved allocation. In [13], the author introduces a model which secure resource allocation in cloud computing, where the author designed fuzzy- logic based trust and reputation model.

Many researches have been done in order to provide efficient method to integrate mobile devices and cloud computing environment. In [14], the author presents concept where cloud computing is utilized in order to improve the capability of mobile devices. In [15], the author did some modification to Hyrax which enables mobile devices to utilize cloud computing platforms. The concept of deploying mobile devices as a provider of resources is presented even though the experiment was not integrated. In [16], the authors only concentrate on the use of partition policies to hold the effect of application on mobile devices. However, they did not resolve any other issues regarding mobile cloud computing or fog computing.

Fog computing technology is still in its early stage and needs more time to develop like cloud computing. To the best of our knowledge, there are not many researches considering collaboration of fog and cloud computing to provide efficient way of delegating IoT services between fog and cloud to better balance workload/requested services/resources. Furthermore, we introduce new methods to delegate services to multiple fog and cloud computing based on linearized decision tree which considers three conditions (service size, completion time, and VMs capacity). Moreover, we introduce new strategy for data distribution and introduce an algorithm to preform resource allocation to guarantee SLA and QoS.

3. Proposed Architecture and Scenario

In this section, we will introduce our new system architecture, explain its component and scenario, and explain the advantages and disadvantages of fog and cloud collaboration.

3.1. System Architecture. Figure 1 illustrates our proposed system architecture which consists of three layers; upper layer, middle layer, and lower layer. Table 1 illustrates our system components and explains their role.

Table 1: System component.

Component	Description
IoT devices	All smart devices that are capable of connecting to internet.
Cloud/fog broker	Responsible for receiving user request/services, providing services/search for VMs, and delegating service to other fog/cloud environments.
Cloud/fog computing server	Responsible for providing requested services/resource, processing them, and delivering them back to broker.
Services monitor server	Responsible for maintaining and storing record of current service and their progress and providing/checking available space for new services.
3rd-party cloud server	Responsible for providing services to fog broker and cloud broker.
VMs occupancy	Responsible for providing list of the current available VMs capacity and showing the used available VMs capacity.
Services map table	Responsible for presenting map of services and their divided chunks in the same and/in other fog computing environments as well as in cloud computing environment.

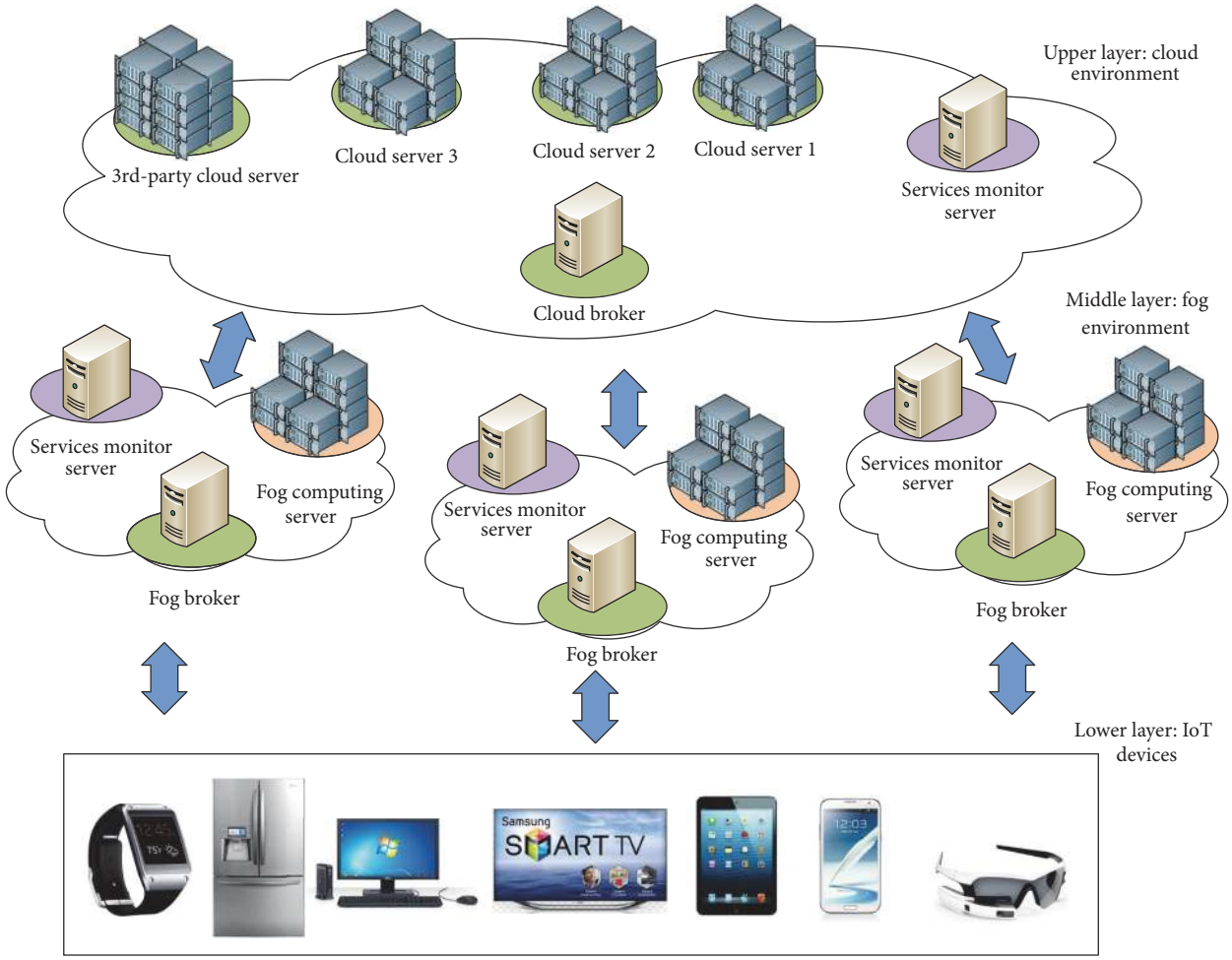


Figure 1: Proposed system architecture.

We provide detailed assumption for our architecture. Our assumption is as follows:

(1) We assume that there are 3 fog computing environments which have closer distance between each other and user smart devices as well as long distance between fog and cloud. Each fog computing environment will include fog broker to manage request services, obtain information of VMs capacity in other fog/cloud environments, and so forth. (Note: in case of larger network area, it is possible to have more than 3 fog computing environments.)

(2) We assume that all services will be requested from fog computing environment. Based on three conditions such as services size, completion time, and VMs capacity, fog broker will decide to process the current requested services in current fog or in other fog/cloud computing environments.

(3) We assume that there are delay-sensitive and nondelay-sensitive requested services, applications, and data.

(4) We assume that all services monitoring server in fog will sync their VM capacity status and current services processing with services monitoring server in cloud. When any fog needs more VM capacity, fog broker will obtain that information from services monitoring server in cloud which will reduce the search time of VMs capacity in other fog or cloud computing environments.

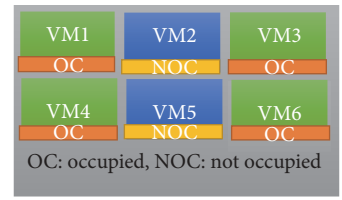
3.1.1. Upper Layer of System Architecture. The upper layer represents cloud computing environment. This layer consists of cloud broker, services monitor server, cloud servers, and 3rd-party cloud server. In case that there is no available capacity in fog to process service and the requested services that is needed later, then the fog can delegate these services to cloud computing.

3.1.2. Middle Layer of System Architecture. The middle layer represents fog computing environment and consists of three fog computing environments. Each fog computing environment consists of fog broker, fog computing server, and services monitor server. Each layer can be aware of each other through using unique communication address for each environment configured by policies or by the cloud itself. Note that it is possible to have more than 3 fog computing environments when we cover larger area/state. All ubiquitous and future services/resources can be requested from any fog environment as well as cloud based on service size (e.g., large or small), requested time (e.g., now or later), and VMs capacity (e.g., occupied or not occupied).

3.1.3. Lower Layer of System Architecture. The lower layer represents user smart devices. User smart devices can consist of smart phone, smart tablets, smart sensors, smart wearable devices, thin-client, smart home appliances, and so forth. Some of these smart devices have different specification and capabilities (e.g., computation process, storage space, screen resolution, and bandwidth). Fog computing can provide efficient way for them to perform these tasks over the Internet in less respond time and efficient performance.

3.1.4. The Role of Services Monitor Server. The services monitor server consists of two important components such as VMs occupancy table and services map table.

The VMs occupancy table is used to keep a list of VMs capacity and occupancy (e.g., occupied and not occupied) in each fog and cloud computing environment. The benefit of this table is to provide fast way for broker to decide on where to process the current service based on service size and the time needed to be completed. In case there is not enough capacity in current VMs at current fog, then the broker can send request to service monitor server in cloud which will store/keep record of all VMs capacity in other fog and cloud environments. Here, the broker can request to reserve these VMs for their current service from other fog/cloud environments. Figure 2 illustrates a sample of VMs occupancy table.



VMs number	Service ID	Fog/cloud	Parts	Progress	Exp_Finish_Time
VM1	1035	Fog 1	2 out of 4	20%	00:00:00
VM6	1035	Fog 2	3 out of 4	40%	00:00:00
VM3	1040	Fog 1	5 out of 6	80%	00:00:00
VM4	1135	Cloud 1	1 out of 3	95%	00:00:00

Figure 2: Illustrating sample of VMs occupancy table.

Services map table is used to keep/store list of currently processed services and their location. For example, sometimes, service is requested from current fog/cloud environment and to complete this service we need the collaboration of 2 or 3 VMs; however, we only have 1 VM at current fog. As a result, the broker will search the cloud for not occupied VMs, reserve them, and request processing the rest of the service in cloud. Here, this table will list service ID, VMs number, location, progress time, expected finishing time, and the IP address which was used to send the services.

3.2. Scenario. As fog computing is gaining more popularity for being near the underlying network and extending cloud computing services/resources near to user location, envision some situation where IoT devices user can benefit from using fog computing environment to resolve any problem, especially when they are in public or at home. We can use the example of many IoT smart devices such as google glass, smart oven, and smart refrigerator. In the first example, the user wants to use their smart oven to cook some food. In order to do so, the smart oven wants to search the Internet to obtain the right temperature which is needed to cook the food. However, the smart oven has limited capability for searching the Internet. As a result, the smart oven can connect to local fog computing environment which in return searches for the right information, and finally, stores that information in the smart oven application. Furthermore, the user can input name of many foods and ask fog to search for the right recipe. Here, the smart oven receives the requested information in convenient and short time.

Another example could be google glass. Let us assume that google glass user is taking pictures which required obtaining information such as sightseeing and cloths. Google glass might have limited capabilities to do searching which

might consume more power and need more resources, big storage, and higher bandwidth. Here, the user can take some pictures of video and send them to fog broker in fog computing for information searching. In case there are many services requested/assigned to current fog, fog broker will collaborate with other fog/cloud computing environments and delegate the services to them. In fog/cloud environment, the service will be divided into many chunks which will be assigned to VMs for processing. After processing is completed, these chunks will be sent to fog/cloud broker where they will be combined and sent back to user devices.

Our proposed scenario illustrates the advantages of utilizing collaboration of work between fog and cloud computing environment. This collaboration efficiently increases the chances of providing efficient method for services delegation, optimizing resources, and optimizing data distribution between fog and cloud computing environment.

3.3. Fog and Cloud Collaboration. In this section, we will introduce the advantages and disadvantages of fog and cloud computing collaboration.

Advantages of fog and cloud computing collaboration are as follows:

(1) Dividing the work load between fog and cloud leads to fast completion of requested tasks when there are many requests (e.g., video, movies, and clips) which are requested at the same time/peak time (e.g., World Cup show and Olympic games events).

(2) The collaboration between fog and cloud lead to better managing of network performance by dividing requested services to small parts and sending them through the network to different fog or cloud for processing. This will reduce the network overload which in return will reduce fog and cloud performance overload.

(3) Fast resources allocation for requested services leads to QoE and efficiently managing resources to a variety of fog and cloud consumers.

(4) Fog and cloud broker can communicate to manage and organize requested services and VMs capacity.

The disadvantages of fog and cloud computing collaboration are as follows:

(1) It might take more time to search for free VMs capacity from other fog or cloud computing environments. To solve this issue, we include in both fog and cloud environments services monitor server which keeps record of current free VMs and VMs status. When fog needs more VMs, fog broker will request VMs capacity of other fog environments from services monitoring server in cloud which will store VMs capacity and currently process services of all fog environments.

(2) Dividing many services to small parts and sending them to other fog or cloud environments might create larger table with larger size when it comes to request certain services that are larger in size.

Table 2: IoT services delegation constrain cases.

	Service size	Completion time	VMs capacity
Case 1	Small	Now	Occupied/not occupied
Case 2	Small	Later	Occupied/not occupied
Case 3	Large	Now	Occupied/not occupied
Case 4	Large	Later	Occupied/not occupied

4. Proposed Mechanism

In this section, we will introduce our methods which we used for IoT services delegation, optimizing resources allocation, and optimizing data distribution. Furthermore, we will provide algorithms and mathematical equations as well.

4.1. Services Delegation Process. In this section, we will explain our method which we used to delegate services to other fog environments and cloud computing environments. The delegation of any services requested from any fog/cloud environment is decision rules of linearized decision trees based on three conditions (service size, completion time, and VMs capacity). The requested services size can be small or large, the requested completion time can be now or later, and VMs capacity can be occupied or not occupied at current fog/cloud environment. We consider 4 cases in Table 2 and provide 2 algorithms that explain these cases process in detail. The cases are shown in Table 2.

Both of Algorithms 1 and 2 aim to find where to delegate the services for processing based on service size (e.g., small or large), services completion time (e.g., now or later), and VMs capacity (e.g., enough or not enough) and in some cases we include services that are in queue (e.g., services in queue, yes or no). As for service size, we can, for example, determine the size based on checking if the size is bigger than 500 mb or not. Moreover, we also aim to manage these services in fog and cloud environment. Figure 4 illustrates sequence flow diagram of any service that is requested from fog environment where there is enough VMs capacity. Figure 5 illustrates sequence flow diagram of services that is requested from fog environment where there is not enough VMs capacity.

4.2. Resource Allocation and Data Distribution Process. Many of formerly presented approaches utilize 1/m/1 model to provide solution to previously mentioned problem. Nevertheless, in our proposal, we utilize 1/m/m/1 for solving the same problem, where (1) refers to cloud broker, (m) refers to many paths, (m) refers to many fog brokers in fog environments, and (1) refers to IoT devices users. In detail, IoT devices will send service request to fog broker in fog environment. Fog broker will divide data into multiple blocks where they will be assigned to certain VMs. Each block will be divided into multiple chunks which will be sent to multiple processor for processing. After receiving the processed data, the processor combines them again into one big data and returns the result to user IoT devices.

By using this method, we reduce/eliminate the burden to the system when we process big data size. Therefore, we guarantee the availability of server in fog or cloud

Input: S_s // service size (small or large), S_{ct} // service completion time (now or later), VM_c // VM capacity (occupied = not enough or not occupied = enough)
Output: service delegation/management location // fog or cloud
(1) **If** (Service Size = **small**) && (Service completion time = **now**) && (VMs capacity = **enough**)
(2) **THEN**
(3) Divide requested services to small chunk
(4) Calculate the required no. of VMs
(5) Assign these chunks to the assigned VMs for processing
(6) **else if** (Service Size = **small**) && (Service completion time = **now**) && (VMs capacity = **not enough**)
(7) **THEN**
(8) Divide requested services to small chunks
(9) Calculate the required no. of VMs
(10) Obtain list of available VMs capacity in other fog/cloud environment from Services Monitor Server in cloud.
(11) Reserved the VMs and assign the chunks to them.
(12) **else if** (Service Size = **small**) && (Service completion time = **later**) && (VMs capacity = **enough**) && (Services in Queue = **no**)
(13) **THEN**
(14) Process the requested service at current location (fog environment)
(15) Divide requested services to small chunks
(16) Calculate the required no. of VMs
(17) Assign these chunks to the assigned VMs for processing
(18) **else if** (Service Size = **small**) && (Service completion time = **later**) && (VMs capacity = **enough**) && (Services in Queue = **yes**)
(19) delegate the requested services to be processed in cloud
(20) Divide requested services to small chunks
(21) Calculate the required no. of VMs
(22) Assign these chunks to the assigned VMs for processing
(23) **end if**
(24) **end if**
(25) **end if**
(26) After completion the processing of all chunks,
(27) **return** the chunks to broker for combining them and send the result to users IoT devices.

Algorithm 1: Finding IoT services delegation/management in fog/cloud based on three conditions (service size, completion time, and VMs capacity) for cases 1 and 2.

environment to process large number of requested services at peak and nonpeak time, guarantee fast respond time, and assure satisfying quality of services (QoS).

Next, we will explain the process of our work which consists of two stages. In stage 1, firstly, we determine the minimum number of VMs needed to do the job and their speed. Secondly, we divided and assigned data based on VMs capacity. In stage 2, firstly, we distribute data which has different capacity to processors. Secondly, after the completion of processing the divided chunks, they will return to cloud/fog broker to combine them and, finally, they will be sent to IoT devices user.

4.2.1. First Stage of Proposed Mechanism. In the first stage, we determine the minimum number of VMs needed to do the job and their speed. Secondly, we divided and assigned data based on VMs capacity.

(A) Determine the Number of VMs and Speed. We use Algorithm 3 to determine the minimum number of VMs which is required to do the job depending on service level agreement (SLA). Furthermore, we use cumulative distribution function (CDF) $F(x)$ time respond which is available in [17]. The minimum number of VMs m keep increasing until $F(x)$ arrive at the desired targeted probability. As a result, we can receive the required m for SLA. Next, we present description of function $F(x)$ and it is as follows:

$$F(x) = \text{Probability}\left(\text{time of response} < x\right) = \begin{cases} 1 - e^{-\mu x} - k\mu e^{-\mu x} x & \text{for } \sigma = m_i - 1 \\ 1 - e^{-\mu x} - k\mu e^{-\mu x(m_i - \sigma)} \left[\dfrac{1 - e^{-\mu x(1 - m_i + \sigma)}}{1 - m_i + \sigma}\right] & \text{for } \sigma \neq m_i - 1, \end{cases} \tag{1}$$

where $\sigma = \lambda/\mu$.

$$k = p(0)\frac{\sigma^{m_i} - \mu}{m_i!} * \frac{m_i}{(m_i - \sigma)}, \qquad P(0) = \left(\sum_{n=0}^{m-1}\frac{\sigma^n}{n!} + \frac{mp^m}{m!\,(m - \sigma)}\right)^{-1}. \tag{2}$$

λ is the arrival rate and μ is the service rate.

Fog computing infrastructure can provide diversity of services to satisfy a large number of SLAs through utilizing unique scheduling methods such as FCFS which is shown in Figure 6. Thus, we are recommending to allocate the VMs into two groups where the first group will be utilized for shared allocation (SA) $m_{\text{shared Allocation}}$ and the second group will be utilized for reserved allocation (RA) $m_{\text{reserved Allocation}}$.

Input: S_s // service size (small or large), S_{ct} // service completion time (now or later), VM_c // VM capacity (occupied = not enough or not occupied = enough)
Output: service delegation/management location // fog or cloud
(1) **If** (Service Size = **Large**) && (Service completion time = **now**) && (VMs capacity = **enough**)
(2) **THEN**
(3) Process the requested service at current location (fog environment)
(4) Divide requested services to small chunks
(5) Calculate the required no. of VMs
(6) Assign these chunks to the assigned VMs for processing
(7) **else if** (Service Size = **Large**) && (Service completion time = **now**) && (VMs capacity = **not enough**)
(8) **THEN**
(9) Divide requested services to small chunks
(10) Calculate the required no. of VMs
(11) Obtain list of available VMs capacity in other fog/cloud environment from Services Monitor Server in cloud.
(10) Reserved the VMs and assign the chunks to VMs for processing
(11) **else if** (Service size = **Large**) && (Service completion time = **later**) && (VMs capacity = **not enough**) && (Services in Queue = **yes**)
(12) **THEN**
(13) this services will be delegated to other fog/cloud environment
(14) Divide requested services to small chunks
(15) Calculate the required no. of VMs
(16) Assign these chunks to assigned VMs for processing.
(17) end if
(18) end if
(19) end if
(20) After completion the processing of all chunks,
(21) **return** the chunks to broker for combining them and send the result to users IoT devices.

Algorithm 2: Finding IoT services delegation/management in fog/Cloud based on three conditions (service size, completion time, and VMs capacity) for cases 3 and 4.

Input:
(1) λ // rate of arrival
(2) μ // rate of service
(3) SLA(x, z) // x: time of response
// z: probability target
Output: m // required minimum no. of VMs
(4) Float $\sigma = \lambda/\mu$
(5) Function determineMinVM (σ, μ, x, z) {
(6) If (σ == (int) σ) m = (int) σ;
(7) Else m = (int)Math.floor(σ) + 1;
(8) While $F(x) \leq z$, m++;
(9) Return m; // required minimum no. of VMs }

Algorithm 3: Determining the number of VMs.

As for shared allocation, the arrival jobs of SLA are merged in a single steamed and served by m VMs.

And, as for reserved allocation, we provide one VM for each arriving job which is shown in Figure 7. Both fog and cloud computing will utilize the model for shared allocation and reserved allocation.

All of the SLAs in shared allocation will have the same CDF of response time and arrival rate $\lambda = \sum_{i=1}^{k} \lambda_i$. Thus, the minimum number of VMs $m_{\text{Shared Allocation}}$ to meet k SLAs is given by

$$m_{\text{Shared Allocation}} = \max\left(m_1, \ldots, m_i, \ldots, m_k\right). \quad (3)$$

Table 3: An example of proposed cases.

Cases	λ_1	x_1, y_1	λ_2	x_1, y_1	m_{Reserved}	m_{Shared}
Case 1	3.9	3, 0.7	3	10	10	11
Case 2	3.9	3, 0.85	3.9	12	12	10

The number of VMs which is required to satisfy SLA_i of user i is referred to as m_i. Let the smallest number of VMs which is required to meet k SLA in reserved allocation be $m_{\text{Shared Allocation}}$. As a result, $m_{\text{Reserved Allocation}}$ is given by

$$m_{\text{Reserved Allocation}} = \sum_{i=1}^{k} m_i. \quad (4)$$

In this case, when more than 1 user request services with the same SLAs, the shared allocation can provide the same or even enhanced performance than reserved allocation ($m_{\text{shared Allocation}} \leq m_{\text{Reserved Allocation}}$). However, in case that SLA_1 and SLA_2 are not the same for shared allocation, then it will be quite difficult to determine whether shared allocation is better than reserved allocation or the opposite. Table 3 will provide example of two cases for shared and reserved allocation.

Note that, in some cases, we have to consider the case where there are services in queue or not yet decided to where to process the requested services (e.g., in fog or in cloud).

Input:
(1) λ_1, λ_2 // rate of arrival
(2) μ // rate of service
(3) SLA_1, SLA_2
(4) E // processing time expectation
Output:
(5) SA, RA //shared and reserved allocation strategy
(6) Function determineAllocStrategy (λ_1, λ_2, SLA_1, SLA_2, E, μ) {
(7) Calculate SLA difference D
(8) Get the corresponding angle α from the SLA difference table
(9) If ($\mu \geq (1/E[T] + \lambda_1)$ && $\mu \geq (1/E[T] + \lambda_2)$)
(10) *If* (Math.asin(λ_2/sqrt ($\lambda_1 * \lambda_1 + \lambda_2 * \lambda_2$)) $\leq \alpha$)
(11) *Return* RA // reserved allocation
(12) *Else*
(13) *Return* SA // shared allocation
(14) *Else*
(15) *Return false* {

Algorithm 4: Determining the allocation strategy.

Table 4: Service level agreement difference (SLA).

D	α
(0, 20)	0
(20, 40)	20
(40, 66)	50
(66, 88)	70

By examining both cases at Table 3, we notice that, in the first case, $m_{\text{Reserved Allocation}}$ has shown better performance than $m_{\text{shared Allocation}}$ and, in the second case, $m_{\text{shared Allocation}}$ has shown better performance than $m_{\text{Reserved Allocation}}$. We are trying to determine the best suitable strategy for shared and reserved allocation for the purpose of satisfying SLA_1 and SLA_2. Moreover, the VMs are able to guarantee the quality of services (QoS). Let the average number of VMs which is needed to meet a given SLA over considered arrival time be E(SLA):

$$E\,(\text{SLA}) = \frac{1}{k}\sum_{0}^{k}\int (k, x, y)\,. \tag{5}$$

Let D refer to the difference between SLA_1 and SLA_2. As a result, D is given by

$$D = \left|E\left(\text{SLA}_1\right) - E\left(\text{SLA}_2\right)\right|. \tag{6}$$

Algorithm 4 illustrates our allocation strategy to satisfy service level agreements (SLA) and quality of services (QoS).

The relationship between D and angle α is explained in Table 4. As illustrated in Table 4, every D is fixed by the changes in arrival time of λ_1, λ_2 in (0, 30) and the average angle of SLA is different for every range.

We state angle α by the next formula:

$$\sin\alpha = \frac{\lambda_2}{\text{sqrt}\left(\lambda_1 * \lambda_1 + \lambda_2 * \lambda_2\right)}. \tag{7}$$

The next step is to discover the speed of VMs in order to guarantee the quality of services (QoS) for every requested service. We also deploy the little law which is explained in [18]:

$$E\,[N] = \frac{p}{(1-p)} \quad \text{where } p = \frac{\lambda}{\mu}. \tag{8}$$

In (8), we refer to the number of jobs in the system by $E[N]$. Equation (9) presents the expectation of processing time:

$$E\,[T] = \frac{E\,[N]}{\lambda} = \frac{p}{\lambda\,(1-p)} = \frac{1}{\mu\,(1-p)} = \frac{1}{\mu-\lambda}. \tag{9}$$

To satisfy the quality of services (QoS), we set the below formula:

$$\mu \geq \frac{1}{E\,[T]} + \lambda. \tag{10}$$

By using this formula, we are able to discover the VMs rate of service. Moreover, we introduce an example below to make it clear. For instance, let us assume that we want $E[T] \leq 10$ second, $\lambda = 1$ job/second, then the VMs rate which is needed is as follows:

$$\mu \geq \frac{1}{10} + 1\frac{11}{10}. \tag{11}$$

(B) Determine VMs Capacity. When the system receives any service, first, we have to find out if we need to process it at current location or delegate it to other fog/cloud computing environments based on the algorithm which we mentioned in Section 4.1. Then, we can determine VMs capacity. In order to determine the VMs capacity, we will sort, divide, and assign data to VMs current Capacity. We also set data priority by utilizing training data to sort out data. As a result, the data with higher priority will be transferred first and the one with lower priority will be transferred last.

We divide data to blocks of different sizes (e.g., bl_1, bl_2, and bl_n). In order to select the best VMs based on their capacities, we utilize greedy algorithm. Finally, the VMs with higher capacity will be assigned to the block with big size.

4.2.2. Second Stage of Proposed Mechanism

(A) Distribution of Data Block Process. We start distributing data block which has different capacities to processors. When we receive data, it will be divided into blocks of data. These blocks will be divided into small size which is known as chunks (e.g., $chk_1, chk_2, \ldots, chk_n$). Every chunk might have different size based on the strength of bandwidth.

Let us denote the chunk in each block by chk_i, the size of chunk by $w(ch_i)$, and the bandwidth between VMs and processor by bw_i. Let $w(ch_i)/b_i$ denote the time it takes to send data (chunk) from VMs to processor. Note that when we consider method of parallelization, the time it takes to send chunks of data to processors should be even:

$$\frac{w(chk_1)}{bw_1} = \frac{w(chk_2)}{bw_2} = \frac{w(chk_3)}{bw_3} = \cdots = \frac{w(chk_i)}{bw_i} = t, \quad (12)$$

$$\text{Set } S = w(\text{block}) = \sum_{i=0}^{n} w(\text{chk}) = t \sum_{i=0}^{n} b_i.$$

Thus,

$$w(chk_i) = t * b_i = \frac{S}{\sum_{i=0}^{n} b_i} * b_i. \quad (13)$$

As it is shown above, we can determine the size of every chunk to adapt it with the bandwidth. The next process is to sort out the processor based on processor capacities. For example, if the chunk of data is bigger, then it will be sent to processor with higher capacity for processing it.

(B) The Merging of Data Block Process. In this section, we explain the process of merging data block after being processed. After the data block is being processed, it will be send back to fog broker in fog environment for merging process. In service monitor server, the services map table will keep a record of these blocks and the location where they were processed which is illustrated in Figure 3.

VMs number	Service ID	Fog/ cloud	Parts number	Progress	Exp_Finish _Time	IP address
VM1	1035	Fog 1	2 out of 4	15%	00:00:00	169.19.16.10
VM2	1035	Fog 2	3 out of 4	45%	00:00:00	169.18.15.11
VM3	1040	Fog 3	5 out of 6	70%	00:00:00	187.96.53.21
VM6	1135	Cloud 1	1 out of 3	95%	00:00:00	190.35.665.35

Figure 3: Illustrating sample of services map table.

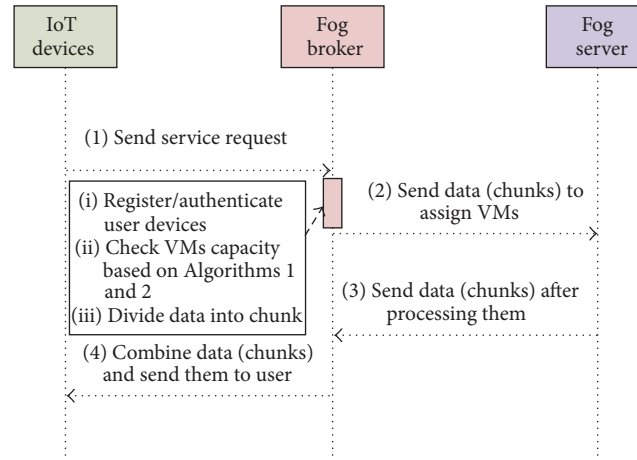


Figure 4: Sequence flow diagram of IoT service process in fog computing environment.

Figures 4 and 5 illustrate the concept where data is divided into chunk and then assigned to VMs in fog or cloud or both of them. Then, after processing these chunk, they are returned to fog broker to merge them and send them to user devices.

Fog computing will act as master which will receive all chunks of data to decrease the complexity that is due to the existing of firewall between processor in fog or cloud environment.

5. Implementation and Analysis

In this section, the numerical experiments results are presented to examine the efficiency of shared allocation (SA) and reserved allocation (RA) as well as comparing our approach's performance with other approaches in terms of processing time to transfer big multimedia data from fog/cloud broker to user smart devices. The comparison method uses one processor [19] to receive data from fog/cloud broker where in our case we use multiple processor.

5.1. Experiment Settings. The characteristics of our target system are illustrated in Table 5. In our PC, we used one Intel Core TM i7 965 and 8 GB RAM. The algorithm was simulated on CloudSim [20]. CloudSim is a framework for modelling and simulation of infrastructures and services in Java jdk-7u7-i586 and Netbeans-7.2.

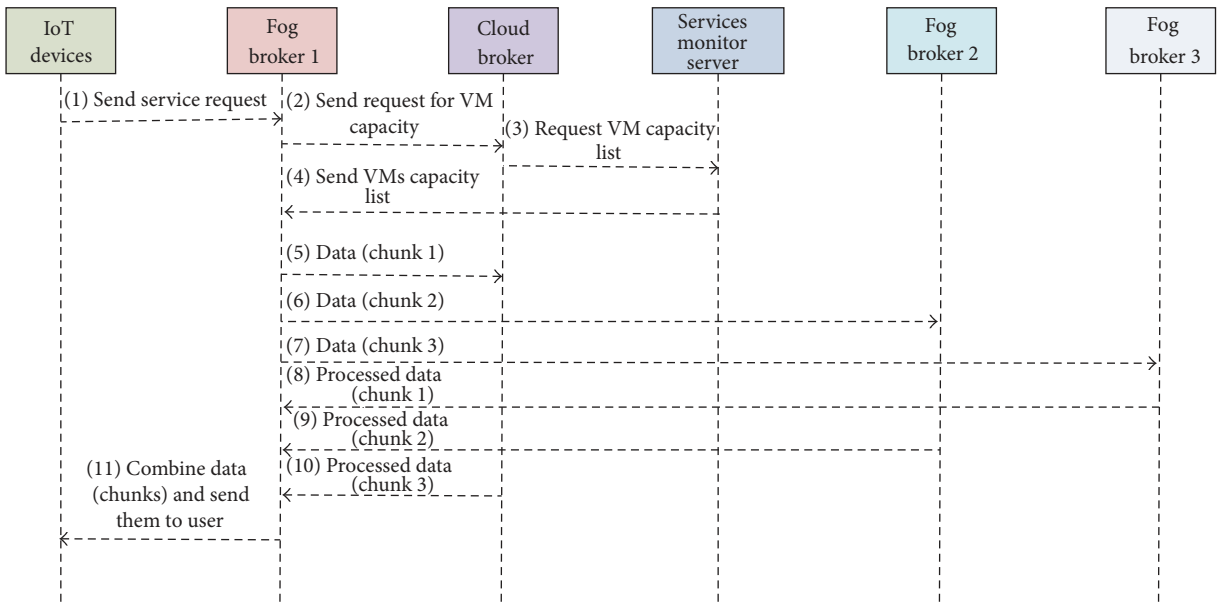


Figure 5: Sequence flow diagram of IoT service which is delegated to other fog/cloud computing environments.

Table 5: Characteristic of the target system.

Parameter	Value
Network	LAN
Topology	Connected
Operating system	Win7 Professional
Number of VMs	25
Number of fog	7
Number of smart devices	10
Bandwidth	[10~512] Mbps

Table 6: Setting for SLA.

Parameter	Value
Response time	[1~10]
Target time	[0.1~0.99]

Table 7: Speeds of requests and response services.

Parameter	Value
Arrival rate	[0.2~3.9]
Service rate	[1~4]

Every parameter in the simulation has different arrival rates λ, response times x, and target probabilities y. Some big files for the abovementioned algorithms are to estimate the required minimum number of VMs for two types of resource allocations and data distribution time. Table 6 illustrates setting for SLA and Table 7 illustrates the speeds of requests and response services.

The experiment result proves that shared allocation and reserved allocation almost have the same impact when SLA is the same for both of them with different arrival rate, response time, and target probability. We did our experiment in the same cases. However, different from other approaches, we used multiple SLA instead of one single SLA.

Figure 8 illustrates different response time of shared and reserved allocation. Our experiment result shows that when the smallest number of VMs decreases, the respond time for shared and reserved allocation increases. In addition, it shows that the probability is almost the same for shared and reserved allocation when we set different response time for shared and reserved allocation.

Figure 9 illustrates SLA different target probability for shared and reserved allocation. Our experiment result shows the minimum number of VMs which is required to meet the satisfaction of SLA. For instance, when the target probability to satisfy SLA is 0.4, we need minimum of 5 VMs for shared and reserved allocation. As a result, it can meet SLA different target probability for shared and reserved allocation.

Figure 10 illustrates different arrival rate of shared and reserved allocation. Our experiment result shows the minimum number of VMs that is required to satisfy SLA which is equivalent to different arrival rate. For instance, when the arrival rate of service is 2, we need minimum number of 3 VMs.

In the case where we consider using multiple SLAs, it is suggested that the strategy of shared and reserved allocation is more resource efficient compared to reserved allocation.

Figure 11 illustrates different SLAs of shared and reserved allocation. The result shows that reserved allocation uses more VMs than shared allocation when number of SLAs

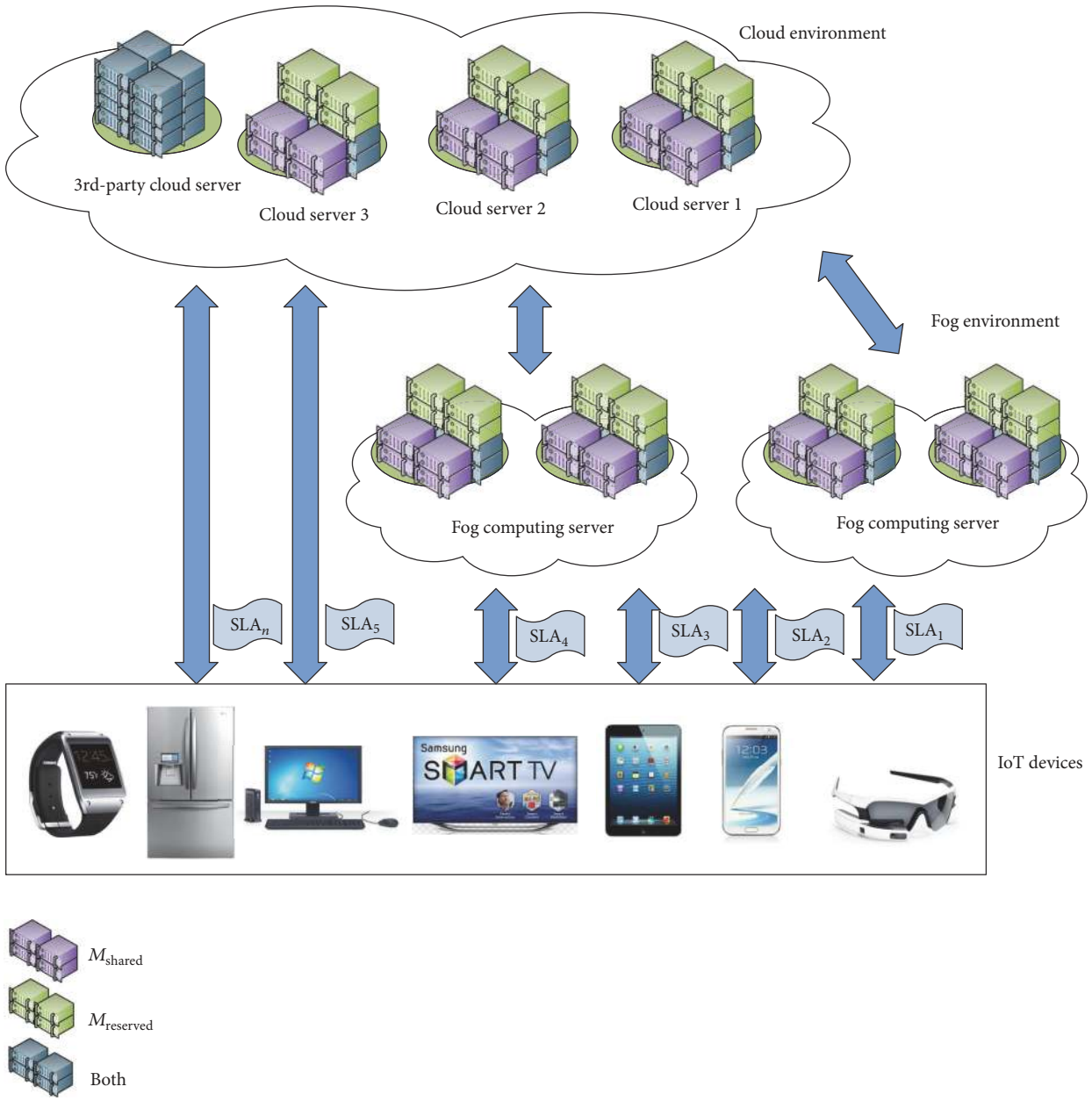


Figure 6: Illustrating our consideration of service level agreement (SLA).

decreases. As a result, reserved allocation can provide guarantee rate due to the offering of resources. For instance, when the number of SLAs is 1, then we need minimum number of less than 5 VMs to do the job. However, when the number of SLAs is 5, then the needed minimum number of VM for shared allocation is 10 VMs and more than 11 VMs for reserved allocation.

A comparison of processing time when sending big size of data to destination for our proposed system with other approaches [19] that utilize one single processor only is illustrated in Figure 12. For instance, by looking at Figure 12, we notice that our proposed approaches generate less processing time than other approaches when we try to send big size of data such as 400 mb. Moreover, our proposed approach shows better performance than other approaches which only use single processor [19]. Other approaches only use one processor where our approach uses multiple processor.

The result, concerning the number of fog/cloud computing environments with respect to IoT devices workload, is presented in Figure 13. We calculate the minimum number of fog/cloud computing environments which is able to satisfy IoT devices workload. The number of fog computing environments increases when the number of IoT devices workload increases and the same thing applies to cloud computing

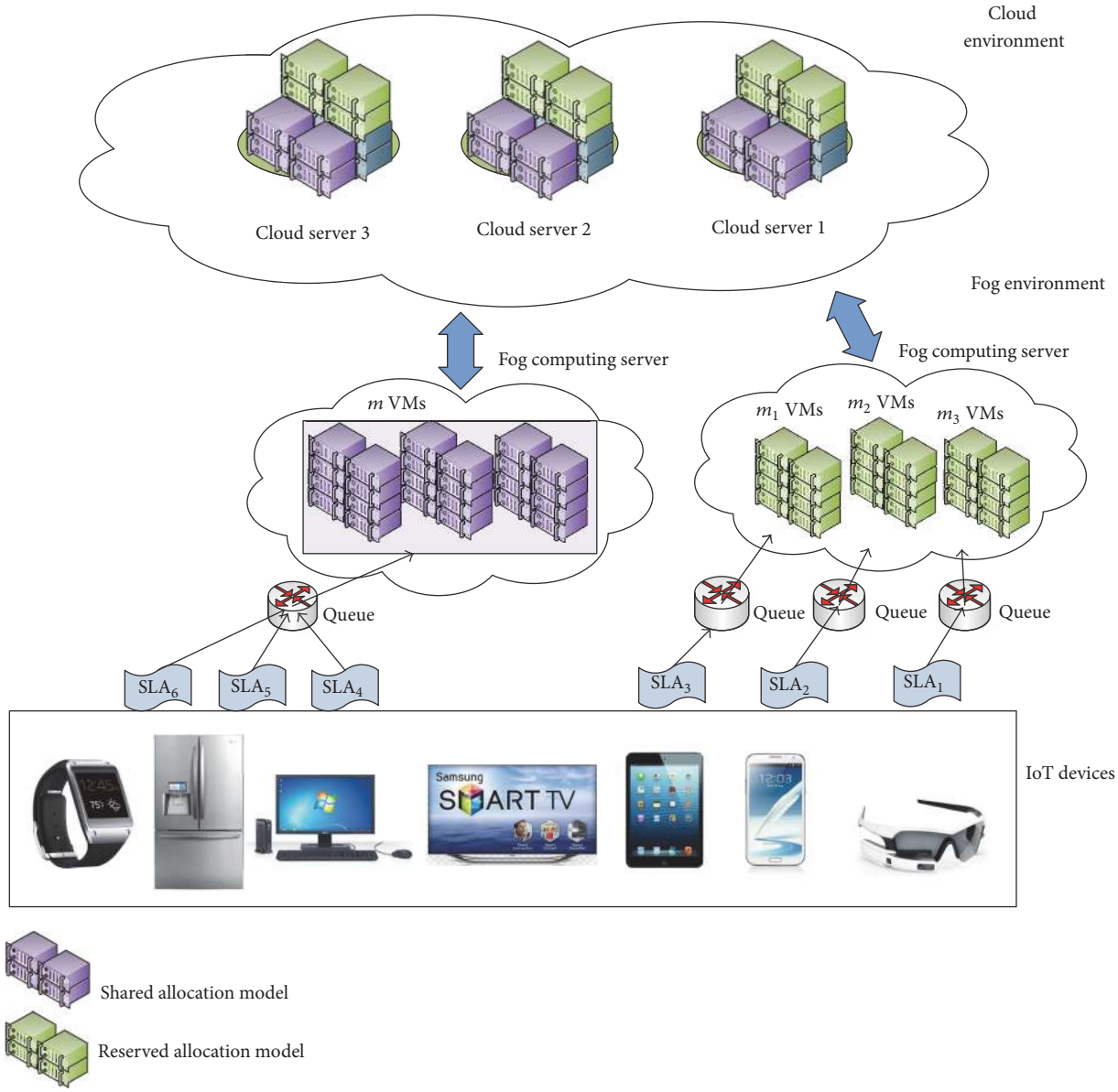


Figure 7: Illustrating our proposed strategy for resources allocation in shared allocation and reserved allocation.

when requested services are delegated to cloud computing. For instance, when the workload of IoT devices is 30 mb, then the minimum number of fog computing environments to satisfy IoT devices increase to 2 fog computing environments.

6. Conclusion

Smart IoT devices are growing rapidly and becoming smarter to access the Internet anytime, anywhere. Nevertheless, smart devices, services, and application are not able to fully benefit from this attractive cloud computing paradigm due to the following issues: (1) smart devices might be lacking in their capacity (e.g., processing, memory, storage, battery, and resource allocation), (2) they may be lacking in their network resources, and (3) the high network latency to centralized server in cloud might not be efficient for delay-sensitive application, services, and resource allocations requests. Moreover, sending or receiving big size of data from centralized server in cloud over the network degraded cloud performance and burden cloud network causing poor QoS, long response delay, and insufficient use of network resources. A localized environment such fog computing can be efficient in resolving the abovementioned issue. In spite of that, the rapid increasing number of services that will be requested from fog computing will generate overhead of services and less services requested from cloud which will result in poor management for both environment and poor QoS.

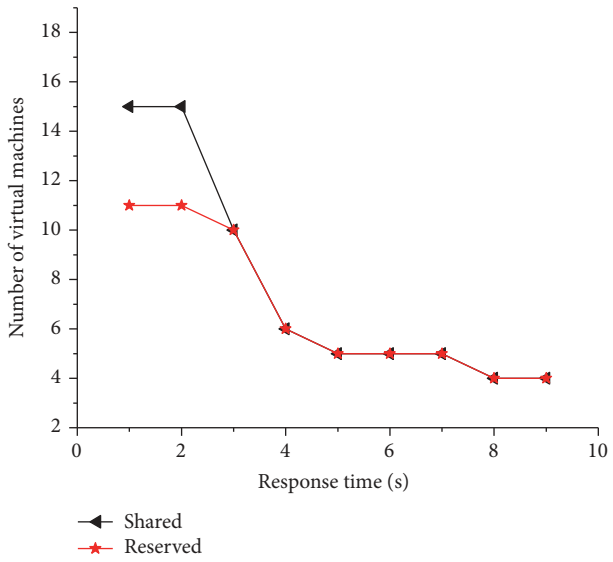


Figure 8: Showing different response time of shared and reserved allocation.

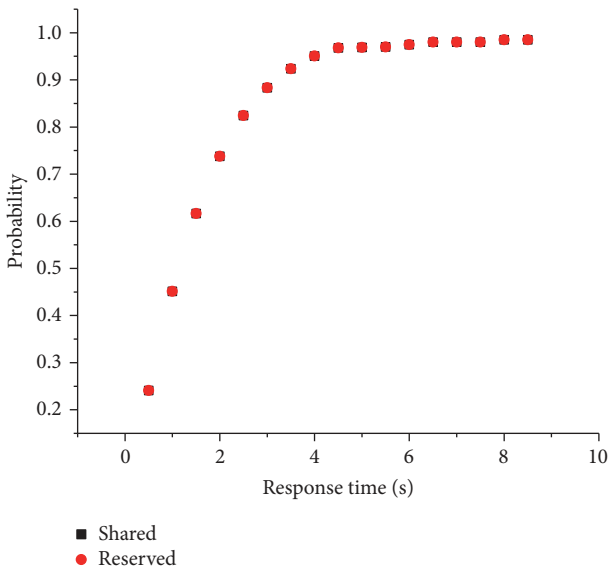


Figure 9: Showing SLA different target probability of shared and reserved allocation.

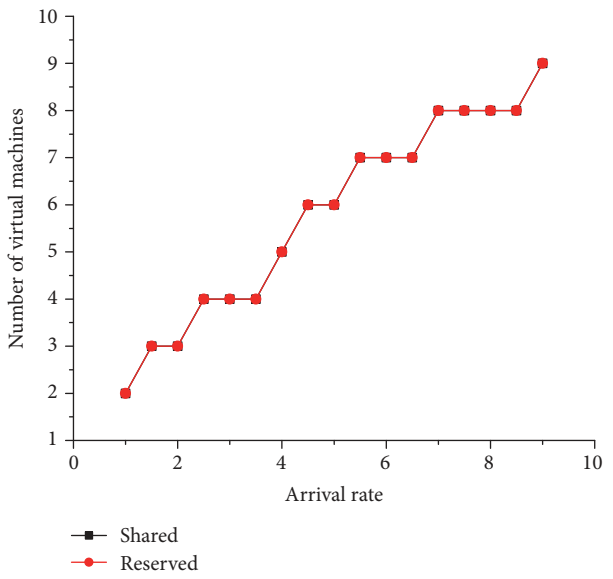


Figure 10: Showing the different arrival rate of shared and reserved allocation.

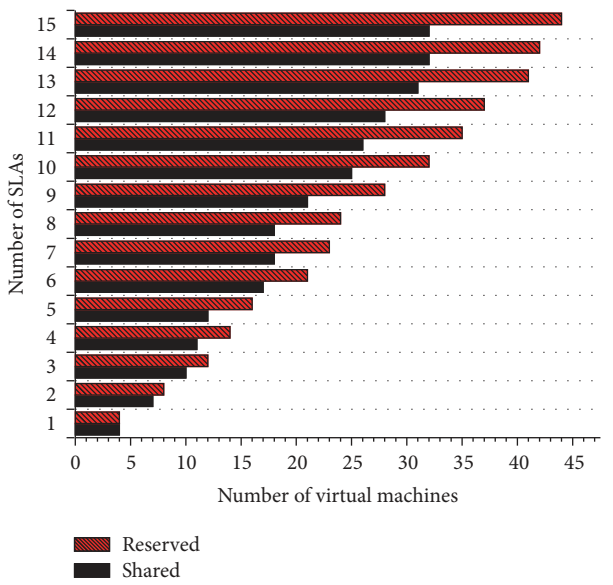


Figure 11: Showing different SLAs of shard and reserved allocation.

As a result, in this paper, we proposed an architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing. We provide new algorithm that is decision rules of linearized decision tree based on three conditions (services size, completion time, and VMs capacity) for managing and delegating user request. Furthermore, we proposed new strategy for optimizing big data distribution in fog and cloud environment. Moreover, we propose algorithm to allocate resources to meet service level agreement (SLA) and QoS. Our simulation result shows that our proposed approach can improve services delegation, management, big data distribution, and resource allocation efficiently and show better performance than other existing methods.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research is supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support Program (IITP-2016-(H8501-16-1015)) supervised by the IITP

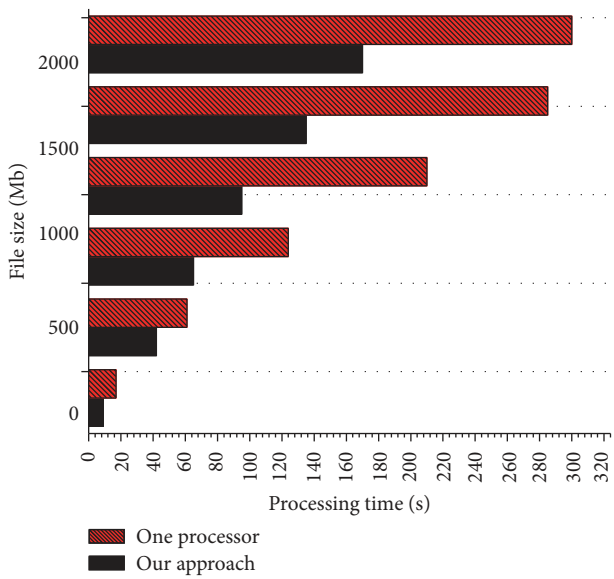


Figure 12: Showing comparison of other approaches (using one processor) with our approach (using multiprocessor).

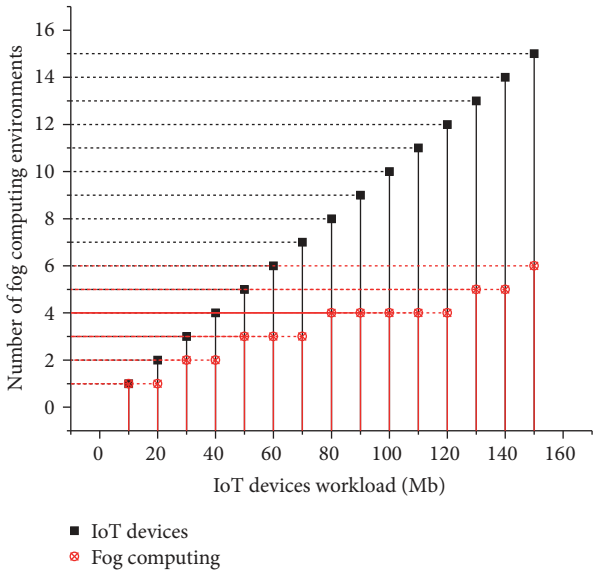


Figure 13: Showing the result of IoT devices workload comparing to the number of fog/cloud computing.

(Institute for Information and Communication Technology Promotion).

References

[1] Y. Pan and N. Hu, "Research on dependability of cloud computing systems," in *Proceedings of the 10th International Conference on Reliability, Maintainability and Safety (ICRMS '14)*, pp. 435–439, IEEE, Guangzhou, China, August 2014.

[2] W. Liu, "Research on cloud computing security problem and strategy," in *Proceedings of the 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet '12)*, pp. 1216–1219, Yichang, China, April 2012.

[3] M. Aazam and E.-N. Huh, "Framework of resource management for intercloud computing," *Mathematical Problems in Engineering*, vol. 2014, Article ID 108286, 9 pages, 2014.

[4] M. Aazam and E. N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *Proceedings of the 12th IEEE International Workshop on Managing Ubiquitous Communication and Services (MUCS '15)*, pp. 105–110, March 2015.

[5] S. Uppoor, M. D. Flouris, and A. Bilas, "Cloud-based synchronization of distributed file system hierarchies," in *Proceedings of the IEEE International Conference on Cluster Computing Workshops and Posters, Cluster*, pp. 1–4, September 2010.

[6] J. Delgado, S. M. Sadjadi, L. Fong, Y. Liu, N. Bobroff, and S. Seelam, "Efficiency assessment of parallel workloads on virtualized resources," in *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC '11)*, pp. 89–96, IEEE, Melbourne, Australia, December 2011.

[7] P. Fan, J. Wang, Z. Zheng, and M. R. Lyu, "Toward optimal deployment of communication-intensive cloud applications," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 460–467, July 2011.

[8] M. Kwok, *Performance analysis of distributed virtual environments [Ph.D. thesis]*, University of Waterloo, Waterloo, Canada, 2006.

[9] G. Y. Jung, N. Gnanasambandam, and T. Mukherjee, "Synchronous parallel processing of big-data analytics services to optimize performance in federated clouds," in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 811–818, IEEE, Honolulu, Hawaii, USA, June 2012.

[10] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," in *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09)*, pp. 101–111, November 2009.

[11] J. Li, J. Chinneck, M. Woodside, and M. Litoiu, "Fast scalable optimization to configure service systems having cost and quality of service constraints," in *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*, pp. 159–168, June 2009.

[12] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? An architectural map of the cloud landscape," in *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pp. 23–31, IEEE, Vancouver, Canada, May 2009.

[13] C. Kamalanathan, S. Valarmathy, and S. Kirubakaran, "Designing a fuzzy-logic based trust and reputation model for secure resource allocation in cloud computing," *The International Arab Journal of Information Technology*, vol. 13, no. 1, pp. 30–37, 2016.

[14] L. Xun, "From augmented reality to augmented computing: a look at cloud-mobile convergence," in *Proceedings of the International Symposium on Ubiquitous Virtual Reality (ISUVR '09)*, pp. 29–32, Gwangju, South Korea, July 2009.

[15] E. E. Marinelli, *Hyrax: cloud computing on mobile devices using MapReduce [M.S. thesis]*, Computer Science Department, CMU, Pittsburgh, Pa, USA, 2009.

[16] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Middleware 2009*, J. M. Bacon and B. F. Cooper, Eds., vol. 5896 of *Lecture Notes in Computer Science*, pp. 83–102, Springer, New York, NY, USA, 2009.

[17] M. Andreolini, S. Casolari, and M. Colajanni, "Autonomic request management algorithms for geographically distributed internet-based systems," in *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '08)*, pp. 171–180, IEEE, Venice, Italy, October 2008.

[18] R. Sheldon, *Introduction to Probability Models*, Elsevier, 10th edition, 2010.

[19] H. C. Gonzalo and D. M. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, no. 6, pp. 1–5, San Francisco, Calif, USA, June 2010.

[20] Cloudsim, "A framework for modeling and simulation of cloud computing infrastructures and services," https://code.google.com/p/cloudsim/downloads/list.