Wireless Communications and Mobile Computing

Mobile Edge Computing

Lead Guest Editor: Robert Hsu
Guest Editors: Shangguang Wang, Anna Kobusinska, and Yan Zhang

Mobile Edge Computing

Mobile Edge Computing

Lead Guest Editor: Robert Hsu
Guest Editors: Shangguang Wang, Anna Kobusinska, and Yan Zhang

Editorial Board

Contents

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7291954, 3 pages
https://doi.org/10.1155/2018/7291954

Editorial

Mobile Edge Computing

Ching-Hsien Hsu,[1] **Shangguang Wang**,[2] **Yan Zhang**,[3] **and Anna Kobusinska**[4]

[1]*School of Mathematics and Big Data, Foshan University, China*
[2]*Beijing University of Posts and Telecommunications, China*
[3]*University of Oslo, Norway*
[4]*Poznan University of Technology, Poland*

Correspondence should be addressed to Ching-Hsien Hsu; chh@chu.edu.tw

Received 13 May 2018; Accepted 13 May 2018; Published 27 June 2018

1. Introduction

The advent of mobile cloud computing has increased expectations for optimal and reliable services and support for mobile users. The large pool of cloud resources and services has enabled the emergence of many novel applications for smart environments. However, the state of the art of mobile cloud computing turns into a problem for communication-intensive applications, which need to meet the delay requirements. The problem becomes even more intense in smart cities or Internet of Things. Current cloud computing paradigm is unable to meet the requirements of low latency, location awareness, and mobility support.

Mobile edge computing (MEC) is emerging as a very promising computation architecture by pushing computation and storage closer to end users with both strategically deployed and opportunistic processing and storage resources. Such mechanism is essentially different from the traditional cloud computing. MEC aims to enable millions of connected mobile devices to execute the real-time applications directly at the network edge. The distinguishing features of MEC are its closeness to end users, mobility support, and dense geographical deployment of the MEC servers.

This special issue aims at presenting the current state-of-the-art research and future trends on various aspects of mobile edge computing techniques for cloud-based IoT applications and attempts to build highly adaptive smart environments that can automatically adapt behaviors to the amount of available resources. The main areas covered by this special issue or main topics cover methodologies, modeling, analysis, and newly introduced applications. Besides the latest research achievements, this special issue also deals with innovative commercial management systems, innovative commercial applications of MEC technology, and experience in applying recent research advances to real-world prob lems.

Papers selected for this special issue represent recent progress in the field, including works on communication technologies, cloud and fog computing, information security, mobile social networks, and machine learning. All of these papers not only provide novel ideas and state-of-the-art techniques in the field, but also stimulate future research in the sustainable environment.

2. Architecture and Resource Management

Recently, the edge computing paradigm has attracted interest from both industry and researchers, carrying the promise of a new communication era in which industry can meet the rising performance needs of future applications. The paper by K. Toczé and S. Nadjm-Tehrani, entitled "A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing", presented terminology and architectures to characterize current works within the field of edge computing. This work reviews a wide range of recent articles and categorizes relevant aspects in terms of resource type, objective of resource management, resource location, and resource use. The authors tried to provide an overview, not from a cloud perspective or an IoT device perspective but with a focus on edge resource management. Taxonomy and analysis of this paper can be used to identify some gaps in the existing research.

The paper by J. Wei et al., entitled "AIMING: Resource Allocation with Latency Awareness for Federated-Cloud Applications", proposed a novel resource allocation approach from the perspective of application providers with different types of resources taken into consideration, aiming to minimize the latency constrained by monetary overhead in the context of federated-cloud. The network resources are deployed and selected according to k-means clustering. The total latency among data-centers is optimized under the constraint of budget based on binary quadratic programming. Experimental results show that the classification is more accurate and effective than graph theory.

Aiming at QoS and user satisfaction, the paper by G. Li et al., entitled "Method of Resource Estimation Based on QoS in Edge Computing", uses weighted Euclidean distance similarity to classify multiple QoS attribute resources and regression-Markov chain prediction method to analyse the change of the load state of the candidate resources. Penalty factor and Grey incidence matrix for improving accuracy of similarity matching were introduced. Effectiveness of the matching method was validated through simulation. The authors proved that regression-Markov chain prediction method can improve the prediction accuracy.

3. Performance Monitoring and Security

In mobile edge computing, edge nodes access resources and services via application servers such as LTE base station and wireless access point. The paper by Z. Wang et al., entitled "Detection Performance of Packet Arrival under Downclocking for Mobile Edge Computing", investigated a novel downclocking technique for low-power WiFi networks. This work theoretically studied the crucial impact of tolerance threshold, correlation threshold, and energy ratio threshold on the packet detection performance. Extensive Monte Carlo experiments show that the proposed theoretical model is with high accuracy. This study can help system developers set reasonable system parameters for WiFi downclocking.

With the development of Internet of Things (IoT), the amount of data transmission shows a trend of exponential increment. The paper by G. Li et al., entitled "Data Processing Delay Optimization in Mobile Edge Computing", proposed a three-layer network model, which combines cloud computing and edge computing. In edge computing layer, a computational scheme of mutual cooperation between the edge devices was presented to reduce the communication delay. In cloud computing layer, a balanced transmission method to solve the data transmission delay from edge devices to cloud servers was introduced. Experimental results show that the proposed architecture can effectively reduce data processing delay and perform better than both the single edge node and traditional cloud computing.

Because there are many attacks against the fog computing network, to effectively handle security threats in fog computing system, the paper by X. An et al., entitled "Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC", proposed a new lightweight intrusion detection system, called Sample Selected Extreme Learning Machine. In the proposed architecture, classifiers are deployed on fog nodes for intrusion detection. Training data sets are stored in the cloud server, so that fog nodes do not need to handle large amounts of data sets. In addition, sample selection process is performed in the training phase in order to improve the classifier algorithm so that it can become more lightweight. Experimental results verified that the proposed architecture performs well in intrusion detection in terms of accuracy and training time.

4. Tools and Applications

Water conservancy engineering presents the national fundamental industry and is vital in national economic development. The construction of water conservancy projects is generally large in scale, with high investment, wide geographical distribution, and decentralized management and in a long construction period and contains a large amount of information. The paper by Y. Fan et al., entitled "Cloud/Fog Computing System Architecture and Key Technologies for South-North Water Transfer Project Safety", proposed a new safety system architecture for water conservancy engineering based on cloud/fog computing. The proposed architecture considered project safety, water quality safety, and human safety. Using IoT devices, fog computing layer was constructed between cloud server and safety detection devices in water conservancy projects. Technologies such as real-time sensing, intelligent processing, and information interconnection were developed. The IoT big data in water conservancy engineering with dense geographical distribution were applied based on multilevel requirements and integrated into a resource integration platform for deployment.

The smart parking system is considered as an important part of the smart city, where many advanced technologies such as the Internet of Things have been ubiquitously deployed in various sectors. The paper by A. Shahzad et al., entitled "Centralized Connectivity for Multiwireless Edge Computing and Cellular Platform: A Smart Vehicle Parking System", aims to manage massive crowd of vehicles and provide better performance for parking searching, reservation, and management. This study comprised several parking points systematically spread over several locations and traceable over the available graphical map, and the overall information is easily accessible using smart devices. The proposed architecture is a novel solution for deploying automated smart vehicle parking system.

5. Conclusions

All of the above papers address either technical issues in communication technologies or information security or propose novel application models in the various cloud/fog and mobile computing fields. They also trigger further related research and technology improvements in application of mobile edge computing. Honorably, this special issue serves as a landmark source for education, information, and reference to professors, researchers, and graduate students interested in updating their knowledge of cloud, mobile edge computing, Internet of Things, and novel application models for future information services and systems.

This special issue of this journal covers different aspects of the problem, from both the theoretical and the practical side. After a large open call, an international editorial committee selected eight research papers. Each paper was reviewed by at least 3 reviewers.

Acknowledgments

The guest editors are deeply indebted to numerous reviewers for their professional effort, insight, and hard work put into commenting on the selected articles that reflect the essence of this special issue. We are grateful to all authors for their contributions and for undertaking two-cycle revisions of their manuscripts, without which this special issue could not have been produced.

Ching-Hsien Hsu
Shangguang Wang
Yan Zhang
Anna Kobusinska

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7476201, 23 pages
https://doi.org/10.1155/2018/7476201

Review Article

A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing

Klervie Toczé and Simin Nadjm-Tehrani

Department of Computer and Information Science, Linköping University, Linköping, Sweden

Correspondence should be addressed to Klervie Toczé; klervie.tocze@liu.se

Received 16 November 2017; Revised 11 March 2018; Accepted 17 April 2018; Published 4 June 2018

Academic Editor: Anna Kobusinska

Edge computing is promoted to meet increasing performance needs of data-driven services using computational and storage resources close to the end devices at the edge of the current network. To achieve higher performance in this new paradigm, one has to consider how to combine the efficiency of resource usage at all three layers of architecture: end devices, edge devices, and the cloud. While cloud capacity is elastically extendable, end devices and edge devices are to various degrees resource-constrained. Hence, an efficient resource management is essential to make edge computing a reality. In this work, we first present terminology and architectures to characterize current works within the field of edge computing. Then, we review a wide range of recent articles and categorize relevant aspects in terms of 4 perspectives: resource type, resource management objective, resource location, and resource use. This taxonomy and the ensuing analysis are used to identify some gaps in the existing research. Among several research gaps, we found that research is less prevalent on data, storage, and energy as a resource and less extensive towards the estimation, discovery, and sharing objectives. As for resource types, the most well-studied resources are computation and communication resources. Our analysis shows that resource management at the edge requires a deeper understanding of how methods applied at different levels and geared towards different resource types interact. Specifically, the impact of mobility and collaboration schemes requiring incentives are expected to be different in edge architectures compared to the classic cloud solutions. Finally, we find that fewer works are dedicated to the study of nonfunctional properties or to quantifying the footprint of resource management techniques, including edge-specific means of migrating data and services.

1. Introduction

Recently, the edge computing paradigm, which consists in having network nodes with computational and storage resources close to the devices (mobile phones, sensors) at the edge of the current network, has attracted interest from both industry and researchers, carrying the promise of a new communication era in which industry can meet the rising performance needs of future applications.

Indeed, with a forecast of 9 billion mobile subscriptions in the world by 2022, of which 90% will include mobile broadband, coupled to an eightfold increase in mobile traffic and 17.6 billion of Internet of Things (IoT) devices also sending data [1], there will be a considerable strain put on the network. The current network technologies need to undergo a paradigm shift in order to handle this situation [2]. Therefore, the aim is to avoid overwhelming the network up to the cloud and, when possible, move some computing and data analysis closer to the users to enable better scalability [3]. Thus, the main idea of edge (or fog) computing is to have intermediate computing facilities between the end devices and the current cloud. As suggested by Mehta et al. [4], this would also enable the current telecom network operators to reduce their operational costs.

In addition to this, moving computing and storage to the edge of the network has other benefits [3] such as reducing the latency and jitter [5], which is especially important for real-time applications such as self-driving cars. Moreover, it enables more privacy for the users by making it possible to keep private data at the edge and enforce privacy policies for the data sent to the cloud (such as blurring sensitive info on a video [2]). Finally, edge networking makes the applications

more resilient by being able to process requests at the edge even if the central cloud is down.

In order to achieve this and to make edge computing a reality and a success, there is a need for an efficient resource management at the edge. Indeed, mobile devices or IoT devices are resource-constrained devices, whereas the cloud has almost unlimited but far away resources. Providing and/or managing the resources at the edge will enable the end device to spare resources (e.g., stored energy in batteries) and speed up computation and allows using resources it does not possess. Moreover, keeping data close to where it was generated enables better control, especially for privacy-related issues. Finally, being located close to the user, edge computing makes it possible to increase the quality of provided services through the use of profiling within a local context, without compromising the privacy or having to handle a large number of users. This is known as *context adaptation*.

Even though this is still an emerging research area, there is a lot of work ongoing under different denominations including mobile cloud computing [6], fog computing [7], edge computing [3], mobile edge computing [8], path computing [9], mobile edge cloud [10], mobile edge network [4], infinite cloud [11], follow-me cloud [12], mobile follow-me cloud [13], multitier cloud federations [14], small cell cloud [15], fast moving personal cloud [16], CONCERT [17], distributed clouds [18], and femtoclouds [19, 20].

Independently of the terminology chosen, which might follow the current naming trend, a common concept here is an intermediate level between the device and the traditional cloud. It is possible to find in the literature numerous surveys about those paradigms in general [6, 10, 21–25], specific aspects of them such as security [8, 26], or specific techniques such as Software-Defined Networking (SDN) [27]. However, those typically do not consider the resource aspect. The existing surveys about resources either consider it at a high level [28] or consider only resource/service provisioning metrics [29].

One area that is of high importance and is present in many use-cases in edge computing is *offloading*. This is the idea of executing a task on a device other than the current execution target. This other device has typically more powerful computational capacities or fewer energy constraints. Resource management is tightly connected to offloading since in order to take a decision to offload, one needs to have knowledge about system resources. This knowledge is provided by resource management techniques. For example, resource discovery can be used as an input for taking an offloading decision, while resource allocation techniques can be used to perform the offloading decision. To the best of our knowledge, existing surveys about resource management for offloading at the edge focus on an end device perspective [30, 31], on the resource allocation part of resource management [32, 33], or on a single-user/multiuser perspective [34].

We aim to complement those surveys by providing a more comprehensive perspective. That is, (a) we consider allocation as one among five resource management objectives, (b) we consider edge resources in addition to end device or cloud resources, (c) we address multiple types of resources and interrelations amongst them, and (d) we review aspects related to locality and what the resource is intended for.

In selecting the survey papers, works considering direct interactions from a device to a cloud [35] or focusing on cloud performance by offloading to the edge [36] are not considered. However, offloading between edge devices or from the edge to the cloud when edge resources are also considered is included. All included papers consider the notion of edge which we attempt to characterize by defining edge-specific architectural instances. This will be done independently of the terminology the authors chose to use. This paper is a substantial extension of our previous much shorter review [37].

In the remaining parts of this paper, we will first present the terminology used, define edge-specific architectures, and present the proposed taxonomy in Section 2. The taxonomy is then exemplified by an extensive review of papers, which are categorized using the taxonomy elements introduced, namely, resource type (Section 3), resource management objective (Section 4), resource location (Section 5), and resource use (Section 6). We then discuss research challenges in Section 7 and conclude the paper in Section 8.

2. Architectures and Research Taxonomy

Edge computing is an innovative area bringing together diverse business sectors such as telecommunication actors, vehicle vendors, cloud providers, and emerging application or device providers, for example, for augmented reality. Therefore, the terminology used in research works is diverse and is still evolving and multiple architectures are considered.

In this section, we present first the relevant terminology associated with edge computing which will be used in the rest of the paper. Then, we discuss the current architectures used and present an architectural breakdown that will be the basis for classifying existing research. Finally, we present our proposed research taxonomy and use it to classify the surveyed works.

2.1. Terminology. Following the development of the IoT, it is nowadays not only computers or smartphones which can be connected to the network but also a large variety of things such as cars, sensors, drones, robots, or home appliances. In this survey, all those objects located at the user end of the network which produce data or need cloud/edge resources will be called *end devices*.

Devices installed at the edge specifically for edge computing purposes are called *edge devices*. We also include under this term the devices that are already now connecting the end devices to the rest of the network, for example, home routers, gateways, access points, or base stations, which are becoming increasingly powerful [38].

Finally, physical components of the cloud are referred to by the term *cloud devices*.

We use those network device classifications to create different *levels* in the network: the device level, the edge level, and the cloud level. Resources that are managed are used to perform *tasks* at some level of the architecture. These can be composed to provide a *service* to the user.

2.2. Current Status of Edge Architectures. There is currently no standard architecture for edge computing, although industry and research initiatives exist, such as the Open Edge Computing (http://openedgecomputing.org/) community, the Open Fog Consortium (https://www.openfogconsortium.org/), and a European Telecommunications Standards Institute (ETSI) standardization group working on Multiaccess Edge Computing (http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing). Current standardization efforts coming from the ETSI group have been reviewed in detail by Mao et al. [34] and Mach and Becvar [33]. Mao et al. [34] also present edge standardization efforts within the 5G standard.

Therefore, current research on edge computing is using several different architectures and there is ongoing work for defining edge computing architectures. Recent surveys focus on presenting these architectures. For example, Liu et al. [10] review different architectures for mobile edge cloud servers and networks, and Mach and Becvar [33] present an overview of proposed solutions enabling computation to be brought close to the end device within the field of mobile edge computing. The approach chosen by Mouradian et al. [39] is to classify the architectures depending on whether they are application-specific or not. They also elaborate on architectural challenges according to 6 criteria including scalability and heterogeneity. Our classification of the device types above is consistent with all the surveys on architecture so far.

2.3. Used Breakdown of Architectures. In this survey, we choose to classify the different architectures into three main categories inspired by the work of Mtibaa et al. [40] and presented in Figure 1. Those categories are technology-independent and aim at visualizing three high-level variants of the edge computing concept that the current works are using.

The first category, named *edge server* and depicted in Figure 1(a), is a generic architecture, where devices are connected to an edge server, which itself is connected to the rest of the network, including the cloud. In this type of architecture, the edge server is at a fixed physical location and has relatively high computational power, though it remains less powerful than a conventional data center used in the cloud computing paradigm. Moreover, there is a clear separation between the device level and the edge level. In the literature, such edge servers are named, for example, cloudlets [41, 42], micro data centers [43, 44], nano data centers [45], or local cloud [46]. They can be located, for example, in shops and enterprises or colocated with the base stations of the telecom access network. Indeed, in the ongoing work on what the fifth generation (5G) of telecommunication networks will look like, a cloud radio access network (C-RAN) is envisaged [47, 48], with connections to other edge computing areas such as mobile cloud computing [49].

The second category, named *coordinator device* and depicted in Figure 1(b), is an architecture, where one end device acts as a coordinator between the other end devices. It also acts as a proxy towards an edge device and/or the cloud if such connectivity is needed. The difference between a coordinator device and an edge server is that the coordinator device can be mobile and has less computational power and bandwidth than an edge server. In this architecture category, the border between the device level and the edge level is not a sharp one, as the coordinator level providing edge functionality is actually an end device. Solutions using this category of architecture are named, for example, fog colonies with a control node [50], vehicular clouds with a cluster head [51], and local clouds with a local resource coordinator [52]. It is interesting to note here that the term *local cloud*, which was already used for describing a part of the edge server architecture category described in the previous paragraph, is used to describe various architectural solutions, illustrating well the fact that the terminology used in edge computing is not yet set.

The last category, named *device cloud* and depicted in Figure 1(c), is an architecture, where the end devices communicate with each other to find needed resources and deliver the wanted services. The devices might communicate with an edge device connected to the cloud if needed but this is not necessary. In this architecture category, the device level and the edge level are thus merged. Research works considering this category of architecture call it opportunistic computing [53], cooperation-based mobile cloud computing [54, 55], or transient clouds [56].

While all these architectures need to be populated with dedicated resource management elements, there is no general agreement about where to place the needed policies. A recent proposal for a generic software architecture that encompasses the edge server version in Figure 1(a) is an enabler for evaluation of multiple resource management policies within common testbeds [57].

2.4. Taxonomy of Edge Resource Management. In addition to classifying the reviewed papers according to the architecture category they consider, we also present a taxonomy of resource management at the edge. This taxonomy, illustrated in Figure 2, aims at getting an overview of state-of-the-art research in this area and presents four main aspects: resource type, objective of resource management, resource location, and resource use.

The two first aspects were constructed by reviewing the current *type* of resources used and the *objective* for which they are used in the literature. The two last aspects are based on mutually exclusive pairs for describing the resource *location* and the *use* of the resource.

In the coming sections, we will describe the different parts of the taxonomy and how the surveyed works can be placed in the four above contexts, as well as the architectural models described.

3. Resource Type

The first step in evaluating the benefit of an edge solution is to decide what are the resource types that can be managed in a better way compared to a centralized system.

An obvious justification for using edge architectures is reducing the response time, which can be done if computation and communication resources are provided and

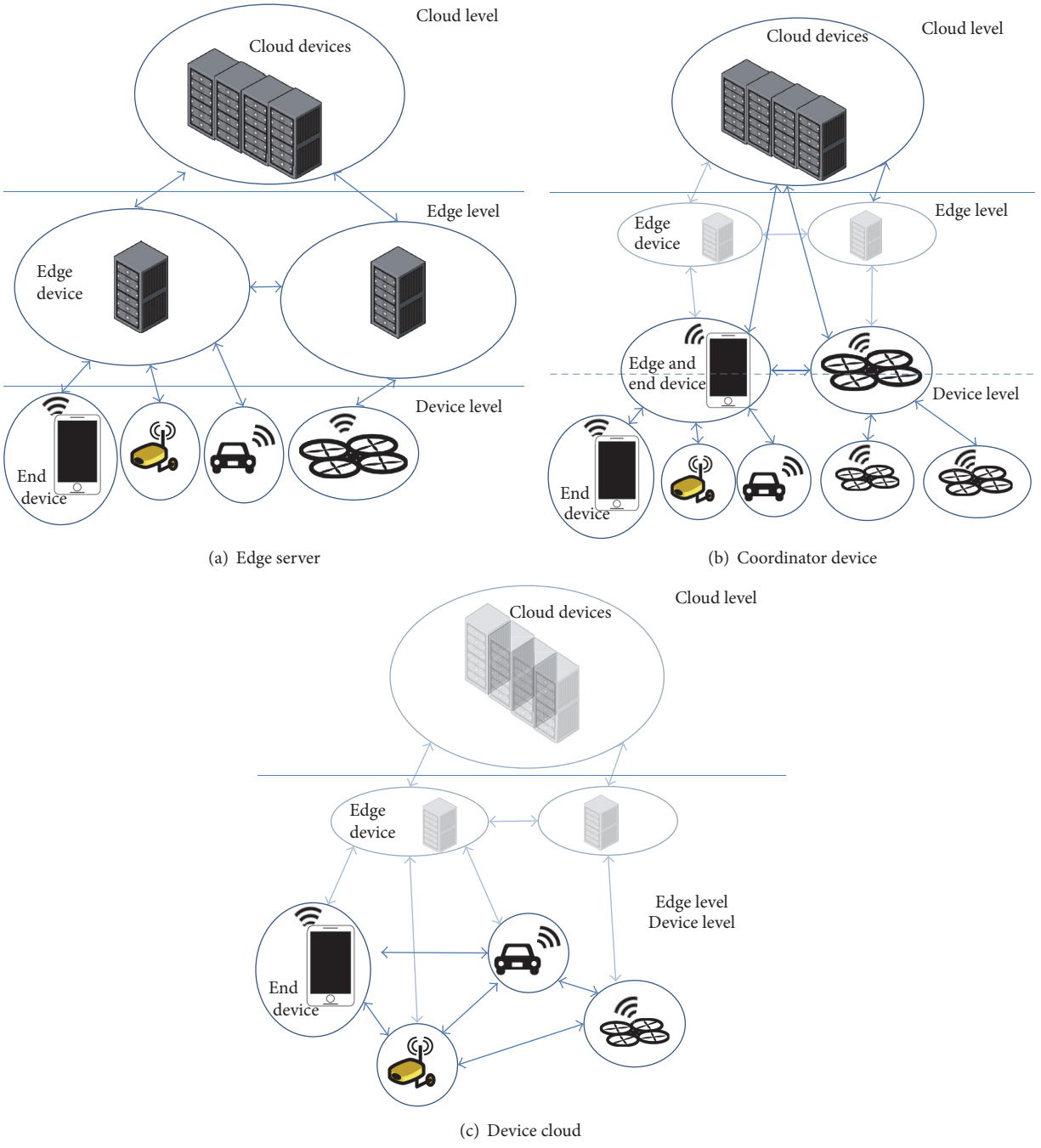

Figure 1: Categories of architectures used in edge computing.

used adequately. Storage as a resource is also a concern, since local storage may benefit security or timeliness due to customized fetching and secure storing mechanisms. A less obvious type of resource is having access to a special type of data (e.g., availability of sensors) that provides local benefits in an application. Examples are the use of cameras or location sensors. The amount and type of data captured in turn affect computation and communication resources (how often to shuffle data and how much to process or filter before shuffling) and implicitly the choice of where and how much of other resources to deploy. The fifth category we consider is energy as a resource, which is clearly influenced by the amount of computation, communication, storage, and data capturing that goes on. Finally, some works consider resources in a generic way using abstract terms such as "Virtual Resource Value" or just as unitless elements in a model.

Table 1 summarizes the surveyed papers in terms of their mapping to the architectural choices in Figure 1. It also shows which resource is focused on within each work, either specific

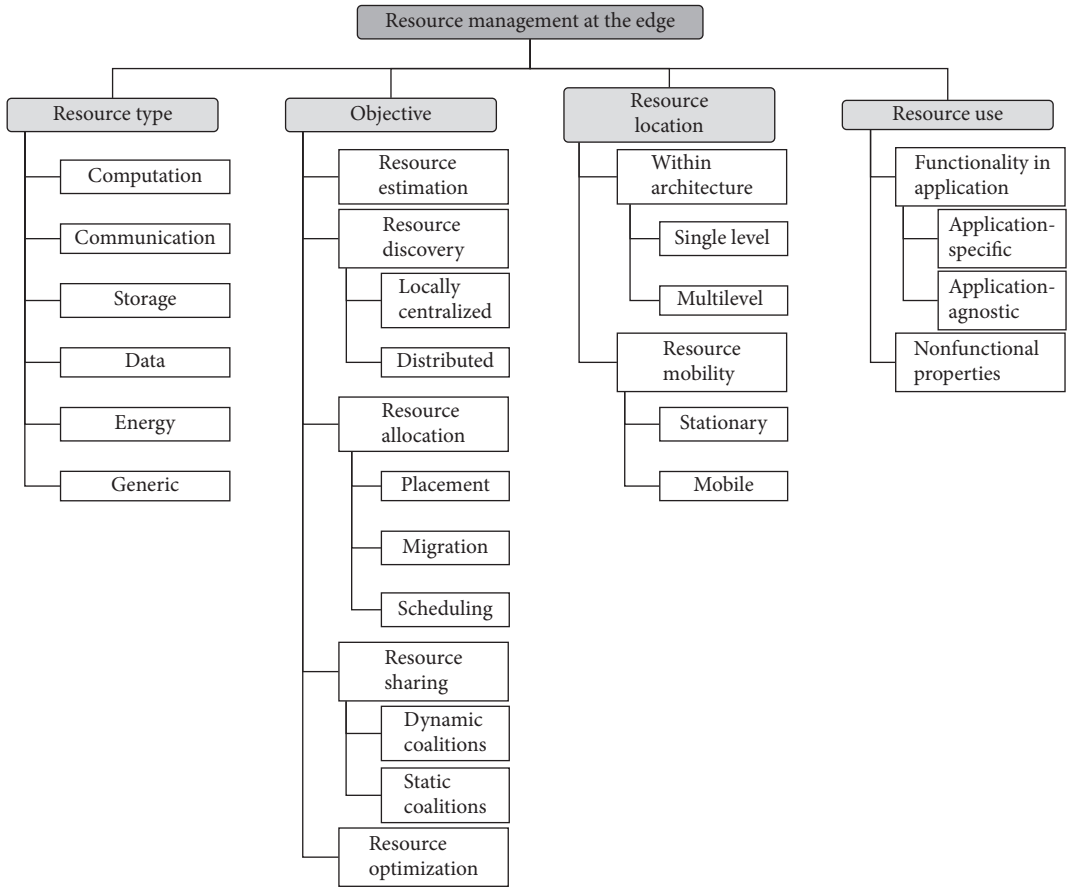

Figure 2: A taxonomy of resource management at the edge.

or generic. As it can be seen, the vast majority of the surveyed articles focus on several resources. Therefore, this section will present the common combinations of resources described above and presented in Figure 2.

3.1. Single Resource Focus. Even though the majority of the surveyed papers choose to focus on several resources, some papers focus on only one resource type. We present those papers in this subsection and then move on to multiresource cases.

3.1.1. Generic. When focusing on a single resource type, most of the works use a generic one, which is used as an abstraction for actual resources.

The abstraction used varies in various articles. For example, Penner et al. [56] work with device capabilities as an abstraction when proposing resource assignment algorithms. Other works, such as Aazam et al. [43, 58], define a new conceptual unit. "Virtual Resource Value" is the unit for any resource, which is then mapped to physical resources according to the type of service and current policies of the cloud service provider.

Sometimes the abstraction is at an even higher level: Wang et al. [77] use generic cost functions that can be used to model many aspects of performance such as monetary cost, service access latency, amount of processing resource consumption, or a combination of these. When proposing a method for online service placement, they, however, analyze its performance for a subset of cost functions related to resource consumption with the claim that this subset is still general.

3.1.2. Energy. Some works focus solely on energy, which is especially important at the edge since devices, in particular end devices, are often resource-constrained. For example, Mtibaa et al. [83] perform offloading between end devices in order to maximize the group lifetime.

Still considering only energy but with another perspective, Borylo et al. [65] classify data centers in two categories (green and brown depending on which source of energy they use) and then use a latency-aware policy to choose a data center for serving a request.

3.1.3. Other. There are works that consider a minimum computational resource unit per device. For example, Fricker et al. [69] use servers as an abstraction (one request occupies one server).

Data as a resource, in addition to sensor data mentioned earlier, can also be seen as content. Gomes et al. [13] propose an algorithm for content migration at the edge, together

Table 1: Surveyed articles according to architecture category from Figure 1 and resource type.

	Article	Computation	Communication	Storage	Data	Energy	Generic
Edge server	Liu et al. [59]	✔	✔				
	Confais et al. [60]		✔	✔			
	Aazam et al. [43]						✔
	Arkian et al. [61]	✔	✔	✔			
	Aazam and Hu [58]						✔
	Fan et al. [62]	✔				✔	
	Oueis [63]	✔	✔			✔	
	Tang et al. [64]	✔	✔				
	Borylo et al. [65]					✔	
	Yousaf and Taleb [48]	✔	✔	✔			
	Wang et al. [49]	✔	✔				
	Gu et al. [66]	✔	✔	✔			
	Tärneberg et al. [67]	✔	✔				
	Plachy et al. [68]	✔	✔				
	Gomes et al. [13]				✔		
	Fricker et al. [69]	✔					
	Rodrigues et al. [70]	✔	✔				
	Zhang et al. [71]	✔			✔		
	Bittencourt et al. [72]	✔	✔				
	Zamani et al. [73]	✔	✔				
	Valancius et al. [45]		✔	✔		✔	
	Chen and Xu [74]	✔	✔			✔	
	Wang et al. [75]	✔	✔	✔			
	Yi et al. [76]	✔	✔				
	Wang et al. [77]						✔
	Sardellitti et al. [78]	✔	✔			✔	
	Singh et al. [44]	✔	✔				
Coordinator device	Nishio et al. [52]	✔	✔		✔	✔	
	Skarlat et al. [50]	✔	✔	✔	✔		
	Borgia et al. [79]		✔		✔		✔
	Athwani and Vidyarthi [80]	✔	✔			✔	
	Arkian et al. [51]	✔	✔	✔			
	Penner et al. [56]						✔
	Bianzino et al. [81]		✔			✔	
	Habak et al. [20]	✔	✔		✔		
Device cloud	Liu et al. [54]					✔	✔
	Mascitti et al. [53]	✔	✔				
	Liu et al. [55]		✔				✔
	Meng et al. [46]	✔	✔				
	Qi et al. [82]					✔	✔
	Mtibaa et al. [83]					✔	

with mobility prediction as an enabler within their new mobile follow-me cloud architecture. This work builds upon the initial follow-me cloud proposal by Taleb and Ksentini [12].

3.2. Multiple Resource Focus. All other surveyed articles are focusing on multiple resource types. In this section, we group the papers according to the different combinations of resources they consider.

3.2.1. Computation and Communication. The most common combination of resource types studied is computational and communicational resources together. Thus, we begin by considering works that study this combination and in one case together with data.

Liu et al. [59] consider wireless bandwidth and computing resource when deciding to handle a request either in a cloudlet or in the cloud. Another example is the work by Bittencourt et al. [72], who consider bandwidth between the

cloud and cloudlet, as well as cloudlet processing capabilities when evaluating different scheduling strategies.

Computational resources can be addressed at a physical level, for example, discussing CPU cycles, or at a conceptual level, for example, use of virtual machines (VMs) as resource elements. In the surveyed articles, Wang et al. [49] consider CPU cycles, Singh et al. [44] consider Millions of Instructions per Second (MIPS), and Rodrigues et al. [70] consider the number of processors per cloudlet. At a conceptual level, Zamani et al. [73] consider different computing resources based on the average number of tasks completed per unit of time, and Plachy et al. [68] allocate computational resources in the form of VMs.

Sometimes the VMs are used as a means to ensure that a task can run given enough underlying resources in the device hosting the VM, for example, in the work by Tärneberg et al. [67].

Instead of using VMs, Yi et al. [76] adopt lightweight OS-level virtualization and a container technique, arguing that resource isolation can be provided at a much lower cost using OS-level virtualization. They also pinpoint that the creation and destruction of container instances are much faster and thus enable the deployment of an edge computing platform with minimal efforts.

As in Section 3.1.3, some works consider a minimum resource unit that corresponds to a device. For example, Meng et al. [46] consider one vehicle as the minimal computing resource unit. Vehicles are aggregated in a resource pool together with communication resources and resource units from the cloud and the edge.

Communication power needed can be considered as a part of the cost when sharing resources [64]. In contrast, communication can be characterized by a delay term impacting the task completion time, like [44, 53, 73].

Finally, Habak et al. [20] consider computation, communication, and data in femtoclouds. The data considered gives information about task dependencies in order to determine in which order the tasks need to be executed and which ones can be run in parallel.

3.2.2. Computation, Communication, and Storage. Other works, in addition to the computation and communication resource types, also include storage in their study.

For example, Arkian et al. [51] tackle resource issues in vehicular clouds by considering all three resource types. Elsewhere, crowdsensing is tackled with the same resource considerations [61].

Another example is the work by Skarlat et al. [50], where they model service demands and a specific kind of resource (sensor data) as well as the computational and storage resources. In this work, communication is considered as a delay term.

VMs can also be considered as an encapsulation of the above three resources in methods that ensure the underlying resources in the device hosting the VM are adequate [66].

Still considering virtualization, Wang et al. [75] propose a system architecture where applications' requests contain computing complexity and storage space requirements. Those requirements are then translated by a SDN controller node into computing power requirements, bandwidth volumes, or requirements on security groups. When trying to allocate more computing and bandwidth resources in an emergency situation, their system will do it by creating new VMs.

Finally, in addition to considering computation, communication, and storage, Yousaf and Taleb [48] emphasize the fact that different resources should not be considered in isolation as there are interactions between them. Thus, they describe and use the concept of resource affinity in their scheme.

3.2.3. Computation, Communication, and Energy. Another combination studied by several of the surveyed articles is computation, communication, and energy resource types.

Athwani and Vidyarthi [80] aim at making resource discovery energy-efficient in order to save battery. Nishio et al. [52] consider energy efficiency in their algorithms but at a more general level, without battery life considerations.

Oueis [63] focuses on energy-efficient communication with the aim of minimizing the communication power needed. Similarly, when studying edge collaboration in ultra-dense small base stations networks with trust considerations, Chen and Xu [74] consider computing (CPU cycles per second), communication as radio-access provisioning, and energy used for both transmission and computation.

Sardellitti et al. [78] propose an algorithmic framework to solve the joint optimization problem of radio and computational resources with the aim of minimizing the overall energy consumption of the users while meeting latency constraints. They first present a solution for the single-user case and then consider the case of offloading with multiple cells in a centralized and a distributed manner.

When considering energy as a resource, a comprehensive discussion of interactions between multiple actions is mapped to energy apportionment policies by Vergara et al. [84]. However, since this work considers edge-/cloud-specific apportionments as one among many application areas, that is, addresses energy sharing in a much wider context, we do not further consider it in our classifications.

3.2.4. Combinations Including Generic Resources. We now consider generic resources in association with other resource types, such as energy or communication.

First, Liu et al. [54] consider abstract tasks and resources to address energy efficiency. They switch between a centralized and a flooding mode depending on energy consumption while keeping the expected value of RIA (Resource Information Availability), which is their quality metric. Qi et al. [82] choose to abstract resources as services and consider energy consumption in the end device when taking an offloading decision.

Regarding communication, Liu et al. [55] use the notion of generic resource (when referring to a combination of bandwidth and CPU available for sharing) as well as concrete bandwidth when nodes are at contact range. Borgia et al. [79] consider data-centric service providers having storage, computing, and networking capabilities but in their evaluation abstract away the storage and computing resources by

only considering the extent to which services are waiting for resources on the provider side.

3.2.5. Other Combinations. Not all works considering computation also consider communication. Less common combinations including computational resources are those with energy and data.

With regard to energy, Fan et al. [62] present a virtual machine migration scheme that aims at using as much green energy as possible in the context of green cloudlet networks.

Data and computation are the focus of Zhang et al. [71] who studied distributed data sharing and processing in order to use data coming from different stakeholders for new IoT applications and propose a new computing paradigm called Firework.

Less common combinations including communication resources include storage and energy resource types.

Confais et al. [60] present how a storage service can be provided for fog/edge infrastructure, based on the InterPlanetary File System, and scale-out network-attached systems. Their aim is to propose a service similar to the Amazon Simple Storage Service solution (https://aws.amazon.com/fr/s3/) for the edge.

Adding energy to storage and communications resources, Valancius et al. [45] consider energy-efficient algorithms when introducing a new distributed data center infrastructure for delivering Internet content and services.

Finally, Bianzino et al. [81] study the trade-off between bandwidth and energy consumption when an end device has access to multiple networking interfaces and can switch between them. They aim at energy efficiency but use an abstract model of power usage based on the amount of data being shuffled.

3.3. Summary of Resource and Architecture Choices. In this section, we have presented the surveyed articles depending on their resource focus. Examining the collection of papers above, resource studies so far seem to focus on computation and communication resources to a greater extent. Moreover, data as a resource is a potential not extensively explored. Similarly, energy is underrepresented among resources studied.

Furthermore, it is noticeable that storage is not the main focus of attention. It could be due to the fact that the cloud is available as a fall-back in many cases. It could also be the case that persistent data storage is not the main focus of most of the applications considered at the edge. Rather, the service or completed task is the main purpose. Another reason could be that presently there are not many critical use cases with latency-constrained storage, but this may change when more and more IoT devices appear in the field. An alternative explanation could be that the authors choose to focus on a reduced set of resources for ease of presentation thinking that the work can be extended to other resources such as storage. Such claims, however, have to be considered with care as this is ignoring the fact that there could be interactions between resources as studied by Yousaf and Taleb [48].

Some resources are dealt with mainly as physical elements, whereas others naturally lend themselves to be defined in abstract ways. For example, sensors are present in the end devices, which can produce useful data needed for the completion of the task (as in [50, 52, 56, 71]), whereas bandwidth (throughput) is a natural abstraction for distinguishing between different radio interfaces or different physical environments (abstracting the impacts of reduced signal strength, interference, etc.).

Moreover, when using a generic resource representation, it is easier to combine several resource types or to combine resources with other performance-related considerations, one example being the generic cost function in the work by Wang et al. [77]. In their performance analysis, they define local and migration resource consumption that can be related, for example, to CPU and bandwidth occupation or the sum of them.

Another point to note is that the first architectural instance (edge server) is the most predominant structure used in the surveyed papers.

4. Objective

A major classification represented in this taxonomy is the objective of resource management. Resource management at the edge can be decomposed into several areas addressing different problems, as shown in the branches under objective in Figure 2. In Table 2, we present which surveyed article addresses which problem(s) and we describe those problems in the following subsections. As it can be seen in the table, one surveyed work can address several of the areas.

The resource management objective is orthogonal to the resource types presented in Section 3 but a discussion of the relationship between objectives of resource management and resource types is conducted in our summary in Section 4.6.

4.1. Resource Estimation. One of the first requirements in resource management is the ability to estimate how many resources will be needed to complete a task or to carry a load. This is important, especially for being able to handle fluctuations in resource demand while maintaining a good quality of service (QoS) for the user. On the supply side, resources at the edge can be mobile and thus become unreachable, which makes them less reliable than resources in a data center. On the demand side, user mobility implies that there can be sudden user churn, with the corresponding dynamic requests having to be handled by the edge.

In their work, Liu et al. [54] use the average of historical data in order to predict the characteristics of resource distribution and usage for the next time slot. The term fog is used by Aazam et al. [43] who propose that it can be used to perform future resource consumption estimation as a first step for allocating resources in advance. They formulate an estimation mechanism that takes into account the reliability of the customer, using what they call the relinquish probability. In another article, Aazam and Huh [58] present the same idea

Table 2: Surveyed articles according to architecture category from Figure 1 and objective of resource management.

		Objective				
		Resource estimation	Resource discovery	Resource allocation	Resource sharing	Resource optimization
Edge server	Liu et al. [59]			✔		✔
	Confais et al. [60]			✔		
	Aazam et al. [43]	✔				
	Arkian et al. [61]			✔		✔
	Aazam and Hu [58]	✔				
	Fan et al. [62]			✔		✔
	Oueis [63]			✔		✔
	Tang et al. [64]				✔	
	Borylo et al. [65]			✔		
	Yousaf and Taleb [48]			✔		✔
	Wang et al. [49]			✔		✔
	Gu et al. [66]			✔		✔
	Tärneberg et al. [67]			✔		✔
	Plachy et al. [68]			✔		
	Gomes et al. [13]			✔		
	Fricker et al. [69]			✔		
	Rodrigues et al. [70]			✔		✔
	Zhang et al. [71]				✔	
	Bittencourt et al. [72]			✔		
	Zamani et al. [73]		✔	✔		✔
	Valancius et al. [45]			✔		✔
	Chen and Xu [74]				✔	
	Wang et al. [75]			✔		
	Yi et al. [76]			✔		✔
	Wang et al. [77]	✔		✔		✔
	Sardellitti et al. [78]			✔		✔
	Singh et al. [44]			✔		
Coordinator device	Nishio et al. [52]				✔	✔
	Skarlat et al. [50]			✔	✔	✔
	Borgia et al. [79]			✔		
	Athwani and Vidyarthi [80]		✔		✔	✔
	Arkian et al. [51]		✔		✔	✔
	Penner et al. [56]			✔		
	Bianzino et al. [81]				✔	✔
	Habak et al. [20]	✔		✔	✔	✔
Device cloud	Liu et al. [54]	✔	✔			✔
	Mascitti et al. [53]			✔		
	Liu et al. [55]				✔	✔
	Meng et al. [46]			✔		✔
	Qi et al. [82]			✔		✔
	Mtibaa et al. [83]	✔		✔	✔	✔

but with an emphasis on how different customers can be charged for the service. Another work by Mtibaa et al. [83] estimates power consumption in order to maximize device lifetime.

Wang et al. [77] use a look-ahead window for prediction into the future in order to minimize cost over time. They study the optimal size for such a window and propose an algorithm using binary search to find this size which they

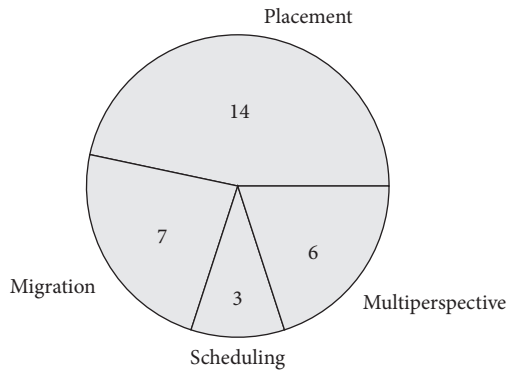

Figure 3: Distribution of resource allocation approaches in the surveyed articles.

evaluate as accurate as it gives results close to the size giving the lowest cost. However, the actual prediction mechanism is assumed to be available.

With respect to computational resources, Habak et al. [20] are estimating the task requirements within a job analyzer. They evaluate the sensitivity of their mechanisms to estimation errors and find that the pipeline job model is insensitive to such errors, whereas the general parallel path model starts exhibiting a significant increase of job completion time if the estimation error variance exceeds 30%.

There are of course many earlier works that use sophisticated prediction mechanisms for estimating future loads in cloud environments (e.g., [85]) but our focus has been on edge-related papers and instances of estimation therein.

4.2. Resource Discovery. As opposed to the estimation problem that relates to the demand side, resource discovery is about the supply side. A management system needs to know which resources are available for use, where they are located, and how long they will be available for use (especially if the resource providing device is moving or it is battery-driven). This area is especially important at the edge, where every resource is not under the control of the system at all times, so the supply is volatile.

The collaboration at the edge can take the form of clusters, as advocated by Athwani and Vidyarthi [80]. They present an algorithm for forming clusters of devices and performing resource discovery within the cluster. Their strategy is that each member of the cluster will inform the cluster head about their available resources and all requests for resources are handled by the cluster head. From their evaluation with respect to energy consumption and delay, they conclude that maintaining the cluster consumes extra energy, especially if the devices are very mobile. Arkian et al. [51] also present a solution using clusters and an algorithm for selecting the cluster head, that is, the vehicle that will be responsible for maintaining the vehicular cloud resources. They use fuzzy logic and a reinforcement learning technique. In order to select the best vehicle, they need to know which vehicle possesses the best communication to the edge node located on the road-side, hence performing resource discovery. This is done in a similar way to earlier work [80]; that is, each potential cluster head node sends a message to the edge node in order to evaluate the link quality before doing the selection. Therefore, those works use a *locally centralized* strategy for resource discovery.

However, using a locally centralized strategy comes at the cost of the necessity to regularly update the node gathering the resource information. Such updates are costly, for example, in terms of energy consumption, as studied by Liu et al. [54]. They propose an algorithm enabling a switch between a locally centralized mode and a *distributed* mode. In the locally centralized mode, end devices propagate their resource information/request to a specific node. In the distributed mode, end devices look for resources in the neighboring devices by using ad hoc WLAN. They qualify their strategy as adaptive as it takes into account the current characteristics of resource distribution and usage in the network. When evaluating the energy consumption of two variants of the adaptive strategy, these perform close to the ideal energy consumption (10% to 13% more energy) and both perform better than strategies using only a distributed or locally centralized mode.

Finally, Zamani et al. [73] use a framework called Comet-Cloud, which performs resource discovery for video analysis and compare the benefit gained to a solution in the cloud.

4.3. Resource Allocation. Resource allocation can be tackled from two different perspectives: *where* to allocate (both initially, but also where and when to perform a migration if needed) and *when and how much* to allocate. Among the dominant approaches to allocation, we find the following three perspectives: placement (14 articles), migration (7 articles), and scheduling (3 articles), as well as a multiperspective one (6 articles) as shown in Figure 3.

In what follows, we group papers that have a single perspective under Sections 4.3.1, 4.3.2, and 4.3.3 and then move on to papers where several perspectives are present.

4.3.1. Placement. Most of the surveyed works emphasize the first perspective, that is, where should the task be executed and the resource allocated for the best possible execution. The definition of best execution varies depending on the considered system and the focus of the research.

Load distribution to achieve lower latency has attracted attention in a number of surveyed works, and it can be seen as an instance of placement. Fricker et al. [69] propose an offloading strategy between edge data centers under high loads that show the benefit of having a larger data center as back-up for a small one. Latency is also the focus of a study by Borylo et al. [65] who investigate dynamic resource provisioning. They present a policy in which the edge can use the cloud in compliance with the latency requirements of the edge but enables better energy efficiency by using resources in data centers powered by green energy.

Also focusing on energy, Mtibaa et al. [83] propose a power balancing algorithm in which a device decides whether to offload and to which other device depending on the energy left in the devices' batteries. In a single-hop scenario, their solution extended the time before the first device of the group runs out of battery by 60% (from 40 minutes to 2 hours) compared to a greedy solution.

Oueis [63] tackles the issue of load distribution and resource allocation in small cell clusters. She formulates a joint computational and communication resource allocation and optimization problem in a multiuser case with a focus on latency and power efficiency. Similarly, Sardellitti et al. [78] study an offloading problem when the end users are separated into two groups: those who need computation offloading and those who do not. They propose a method to jointly optimize communication and computation resources, where both user groups compete for communication resources but only the first group competes for computation resources. They first present an algorithm for the single-user case and then two algorithms for the multiple cells case, a centralized one, and a distributed version to mitigate the communication overhead induced by the centralized approach.

Valancius et al. [45] propose a content placement strategy, where the content is movies. The focus is first on finding the optimal number of replicas of the data to be stored and then on placing the replicas on available gateways. Similarly, Qi et al. [82] present an allocation scheme, where the resource (coming from either a cloudlet or a cloud) is chosen for each task. The aim is to pick the resource from the most suitable location when the user is moving.

Wang et al. [77] study service placement in a system composed of edge server nodes and traditional cloud nodes. Simulation results with real-world traces from San Francisco taxis show that the proposed approach is close to the case of online placement when the future is known, outperforming edge-only or cloud-only solutions. Similarly, Skarlat et al. [50] present a service placement problem for IoT services.

Mascitti et al. [53] present an algorithm where an end device can choose to either use a resource from a node it is directly in contact with or compose different resources from different nodes in order to complete its task. Coming from the same research group, Borgia et al. [79] present a framework where the decision is taken to obtain a service from a local group of end devices or from the cloud, based on an estimation of the time required to obtain the service.

Confais et al. [60] propose a storage mechanism where the objects to be stored are primarily placed locally at their creation but can then be copied to another location if another edge site is requiring access to it. In vehicular networks where resources are mobile, placement has to take account of changes in location. Meng et al. [46] present a resource allocation scheme that can manage resources from a vehicular cloud, the edge, or the cloud. Their focus is on minimizing delays for the users.

In the application domain of healthcare, Gu et al. [66] include VM placement in their optimization problem for analyzing data. Their two-phase solution has a nearly optimal solution and outperforms a greedy strategy with regard to cost.

4.3.2. Migration. Still considering where the task should be executed, when it comes to virtual entities such as services, applications, tasks, and VMs, the focus could also be on how they can be moved during execution if the new location is better, that is, on migration.

For example, Tärneberg et al. [67] study application-driven placement and present a system model for mobile cloud network with a dynamic placement algorithm that guarantees application performance, minimizes cost, and tackles resource asymmetry problems. Plachy et al. [68] propose a cooperative and dynamic VM placement algorithm associated with another cooperative algorithm for selecting a suitable communication path. They use VM migration to solve user mobility problems.

Other works focus specifically on the problem of migrating resources. For example, Gomes et al. [13] present a content-relocation algorithm for migrating the content of caches present in edge devices. This needs a prediction of user mobility.

With respect to virtual computational resources, Fan et al. [62] and Rodrigues et al. [70] focus on VM migration but with different optimization objectives (increase the use of green energy and minimize delays, resp.). Yousaf and Taleb [48] propose a VM migration (and VM management) system that takes into account the relationship between resource units when making migration decisions. They present this work in the context of 5G but it should be applicable to all physical machines hosting VMs.

Finally, Penner et al. [56] introduce the term *transient cloud* and concentrate on task assignment towards a given node. They present a collaborative computing platform that allows nearby devices to form an ad hoc network and provide the ability to balance the assignments among themselves. This may be considered as a form of migration.

4.3.3. Scheduling. While there is a huge body of researches available on when and how many resources to allocate within networking and cloud-specific areas, our goal here was to identify examples where scheduling decisions are at the edge level in the sense of our terminology in Section 2.1.

Regarding when to allocate resources, Bittencourt et al. [72] study the impact of three different fog scheduling strategies on application QoS (concurrent, first come-first served, and delay-priority). For two applications studied, when more than four users are moved between cloudlets, the concurrent strategy exhibits a lot longer delays than the

sequential ones. However, this same strategy is using the network a lot less than the other two.

With the same focus, Singh et al. [44] consider only scheduling for tasks with a *private* tag. Those can only be executed on the local edge server and will be rejected if not enough resources are available. In their algorithm, tasks are considered in an earliest-deadline-first manner.

Regarding how many resources to allocate, Wang et al. [49] propose a joint cost-effective resource allocation between the mobile cloud computing infrastructures and the cloud radio access network infrastructure. If the need of the application is greater than the available computational resources, then they reduce the amount given to each virtual machine so that it fits the total amount available and adapt the data rate accordingly. They show that joint optimization with respect to cost and energy performs better compared to separate cost- and energy-optimization strategies.

Similarly, Wang et al. [75] propose elastic resource allocation for video-surveillance systems. The elasticity comes from an algorithm they propose to handle some emergency surveillance event (like tracking a criminal) that requires a sudden increase of computation and communication resources to make sure that all the possible images are analyzed within a reasonable timeframe. When such an emergency event happens, network bandwidth allocation is reconfigured and computing resources are reallocated (by launching new VMs in the impacted zone and balancing the workload on nodes). When experimenting in their physical testbed, they verified that data propagation round-trip time is about 5 times lower with edge nodes close to the cameras compared to the cloud. They also found that the time for launching new VMs in the emergency mode is between one and two minutes, which they claim is acceptable in such a scenario.

When addressing scheduling, the surveyed articles most often do it at the same time as placement or migration, which is the topic of the next subsection.

4.3.4. Multiple Perspectives. A work that tackles both perspectives is by Liu et al. [59]. It presents a multiresource allocation system that first decides whether the request should be served or rejected (admission control) and then where to run it (edge or cloud level) and finally how much bandwidth and computing resources should be allocated for this task. To do that, they use Semi-Markov Decision Processes and their aim is to maximize system benefit while guaranteeing QoS for the users. To measure the benefit, they use blocking probability and service time as metrics. When evaluating, they compare to two greedy strategies and show that their proposal outperforms the first one and provides a 90% reduction of the blocking probability with only a slight increase of service time compared to the second greedy strategy, which would be acceptable for congested situations.

In the context of video analysis, Zamani et al. [73] also studied those two perspectives. Their scheduling is based on identified chunks of video, applying two alternatives: minimizing computation time or minimizing computation costs. Their placement is done after resource discovery using CometCloud. In their evaluation, they showed that the solution using edge accepts more tasks and in particular more high-value tasks than a solution using only the cloud. Hence, the overall value obtained from the processed data is maximized at the same time as the throughput of the infrastructure.

Also in the area of video analytics, Yi et al. [76] investigate three task prioritizing schemes for scheduling the task requests at a receiving edge node. Their solution, using the flow job shop model and applying a well-known approach (Johnson's rule), aims at minimizing the makespan. Their simulations compared the approach with other strategies (Short IO First, Longest CPU Last) and found that response time was improved. Their work also includes a second perspective, by investigating three task placement schemes for collaboration within the edge level (Shortest Transmission Time First, Shortest Queue Length First, and Shortest Scheduling Latency First). Using their testbed, they found that the Shortest Scheduling Latency First achieves the best performance in terms of task completion time.

Singh et al. [44] consider both placement and scheduling with respect to semiprivate or public tasks (in addition to what was mentioned in the last subsection for private tasks). Those tasks are placed after a decision is taken for the private ones. Still considering Earliest Deadline First, the placement strategy is to try first one's own edge and then one's own cloud and if they are overloaded go to some external edge and then to an external cloud. In the evaluation, they show that, for tasks having tight deadlines, their system RT-SANE will complete a lot more tasks before their deadline than a cloud-only solution.

How many resources are to be allocated to a given IoT data generator is a topic of discussion by Arkian et al. [61], in which they first mathematically model deployment and communication costs on various fog nodes and then decide on placement of VMs to achieve lowest costs. Analyzing monetary costs for compute nodes, their fog solution decreased the cost by over 33% compared to using a cloud solution. For routing and storage monetary cost, the decrease is about 20%.

Habak et al. [20] first consider placement for deciding in which end device a task will be run. They use a path-based assignment policy with the aim of minimizing the overhead of transmitting data needed for task execution between end devices. In the evaluation, this translates into performing better than two other baseline solutions in terms of service completion time. Then, they also consider scheduling of the computation resource. This should be done in a predictable way so that the part of the system distributing the tasks can make good decisions. They propose a fair queuing based task pick-up that ensures a fair execution of the tasks belonging to different services. Moreover, they implement an early pick-up mechanism to enhance the previous mechanism so that a task with an urgent deadline but belonging to a service with a lower priority can be executed before a higher priority task if this one still meets its deadline.

While this is not the focus of the survey and as such is not included in the table, Dong et al. [86] study offloading and Earliest Deadline First scheduling within end devices. They find that one of their proposed approaches maintains good predictability for twice higher CPU utilization than

widely used approaches, while keeping energy consumption reasonable.

4.4. Resource Sharing. Resources on end devices are heterogeneous and most of the time scarce, and edge devices also have limited resources compared to (almost infinite) resources in the cloud. Sharing resources between devices or between end and edge devices aims at tackling three different issues: not having the needed resource at all in the device where the task is initiated, not having enough of it, or using other devices' resources in order to get a faster completion of the task.

Sharing resources is typically realized by pooling resources in the local vicinity of client nodes. This can extend to the edge domain (clustering edge servers) or remain at end devices. The latter is investigated by Skarlat et al. in so-called *fog colonies* [50], by Arkian et al. within vehicular clusters [51], or by Bianzino et al. [81] for uploading data streams in presence of mobility.

We can classify the surveyed articles into two categories according to whether they include how to form the groups of devices that will share resources or if they assume that the formation is already done and focus on the actual sharing. We call these two categories *dynamic coalitions* and *static coalitions*, respectively.

Starting with dynamic coalitions, Chen and Xu [74] and Bianzino et al. [81] include the formation of device coalitions. Chen and Xu [74] do it using a coalition game incorporating trust considerations. When the supply matches the demand, they found that using a coalition can lead up to 40% lower weighted cost (including latency and monetary considerations) compared to a noncooperative scenario. When there is overload or light workload, it is either not possible or not needed to collaborate and the gain is very low. Bianzino et al. [81] express resource sharing as an optimization problem, where the aim is to create as few and large groups as possible to minimize the number of high-energy interfaces that will be used. They evaluate that their algorithm leads to over 60% energy saving of the total energy consumed by the end devices.

Still using dynamic coalitions, Arkian et al. [51] and Athwani and Vidyarthi [80] propose methods to create clusters. The former compare their method to an earlier baseline and achieve 3 times lower service discovery delay and 4,5 times lower service consumption delay for a small number (50) of vehicles. The latter show that energy consumption is similar to a centralized approach, while the delay is closer to a flooding approach (i.e., low in both cases).

However, creating and maintaining a group of devices that can share their resources has a cost, for example, shown by Athwani and Vidyarthi [80] who concluded that maintaining the cluster consumes extra energy, especially if the devices are very mobile. This is why it is beneficial to do the resource sharing in two phases, where the first phase is deciding whether the device gains more by working alone or joining a coalition, and the second one is deciding if the device will consume others' resources [81, 87]. Yu et al. [87] show that their cooperative solution improves user QoS (defined by how much computing and how many bandwidth resources are allocated to a user) by 75%. However, this paper is using traditional cloud resources and not edge, so it is not included in the tables.

Moving to static coalitions, Skarlat et al. [50] consider resources shared between two neighbor fog colonies and achieve a 35% reduction of execution cost compared to a cloud-only strategy. Regarding data, a resource that many stakeholders may be interested in sharing, Zhang et al. [71] present a data sharing framework called Firework. They include two case studies, including the search for a person with the help of multiple cameras from different owners.

Some researchers, such as Liu et al. [55], try to exploit opportunistic contacts between the devices, creating a resource sharing mechanism that enables faster task completion. They propose different models for calculating task latencies and their approximation algorithm performs better than two other strategies. Similarly, Mtibaa et al. [83] define three mobile device clusters (one hop, two hops, and opportunistic) that can share their resources. Their aim is to share resources in order to get the longest possible network lifetime, that is, saving as much energy as possible through offloading to another device so that the devices can stay on longer. They identify two important topological factors: number of hops and disconnection rate due to mobility.

Resource sharing can perhaps speed up the execution of a task, but Nishio et al. [52] argue that this is not bringing any advantage for the user if we do not consider task dependencies in order to provide a service to the user. They provide the example of a GPS service: if the best route calculation is very fast but the downloading of the map is not, the service to the user will not get faster as both are needed. Habak et al. [20] consider sharing of end device resources belonging to a femtocloud in order to execute tasks. In their system, the owner of the end device can configure how they want to share resources via their personalized resource sharing policies.

Finally, even if resource sharing can bring benefits for a group of end devices, it is not obvious that users will agree to share their resources, especially if they are always on the providing side. Therefore there is a need to develop incentives for resource sharing such as works by Tang et al. [64], Bianzino et al. [81], and Chen and Xu [74]. The following mechanisms are provided in the above works, respectively:

(a) A double bidding mechanism for demander and supplier of resources where the focus is on how to encourage mobiles with resources to share them

(b) A mechanism for lending energy to vicinity nodes which is rewarded and can be used in future scenarios when the lending node itself needs energy

(c) Payment incentives for lending out resources

On the same topic, Habak et al. [20] performed a pilot study to identify effective incentive mechanisms. They studied the willingness of around 50 students to share their resources in 4 scenarios and found out that they would agree to share their resources if they are getting compensation (e.g., money) for it or if the reason for the computation taking place is significant (e.g., emergencies).

4.5. Resource Optimization. A fifth objective pursued in the surveyed works is to optimize the resource use at the edge. This is usually a joint objective together with one of the previously described objectives. Which aspect should be optimized and the associated constraints vary among the surveyed works but the three main ones are QoS (often understood as latency), energy, and operational cost. How the optimization problem is formulated and solved also varies, and we present those variations in this section.

First, some articles consider selecting the optimum solution by comparing the results from different candidates and selecting the minimum/maximum value depending on the objective. For example, Yousaf and Taleb [48] select the value maximizing the resource utilization, Athwani and Vidyarthi [80] use the minimum value of a custom function to select the cluster head, and Mtibaa et al. [83] select the configuration maximizing the estimated remaining energy.

Another group of works solve their optimization problem using linear programming [45, 59] or an approximation based on linear programming [55].

A third group of works use integer linear programming [50, 81] or mixed-integer linear programming [62]. Qi et al. [82] formulate their task allocation problem using integer programming and solve it by a self-adaptive learning particle swarm optimization algorithm. First formulating using mixed-integer nonlinear programming, Arkian et al. [61] then linearize the problem and solve it using mixed-integer linear programming. Gu et al. [66] do the same and then use heuristics. Using a different approach, Yi et al. [76] first formulate a mixed-integer nonlinear programming problem but then relax the integer constraints and use sequential quadratic programming for solving.

Some works focus on convex problems, like Wang et al. [77] who use an approximation algorithm in the online case and Nishio et al. [52] who use a heuristic. Starting with nonconvex problems, Oueis [63] casts them into convex ones and Wang et al. [49] first use a Weighted Minimum Mean Square Error-based method on their nonconvex problem to obtain a convex problem that they apply the block coordinate descent method to for solving. Finally, Sardellitti et al. [78] have an optimization problem in the multiple-cells case which is nonconvex and they solve it by developing a method based on Successive Convex Approximation for the centralized approach. For the distributed approach, they choose the approximation functions in a way that allows decomposition in smaller subproblems solvable in parallel.

A further group of works propose their own algorithm or heuristic. Tärneberg et al. [67] approximate an exhaustive search approach yielding an optimal solution but having exponential computation complexity with an iterative local search algorithm finding a local optimal solution. Zamani et al. [73] implement an optimization strategy where constraints on computation time and cost are enforced using an admission control strategy. Wang et al. [77] present a binary search algorithm for finding the optimal look-ahead window size, and Habak et al. [20] propose an algorithm in order to do deadline-based optimization when a helper has to handle multiple tasks belonging to different services. Finally, Liu et al. [54] propose a heuristic algorithm that uses different statistics to estimate the energy that is going to be consumed in each of the two possible modes during a time slot and chooses which mode to use depending on this and other parameters.

Other methods can be used to compare heuristics with baselines or to solve a formulation in a custom form. In the offline case, Wang et al. [77] show that their problem is equivalent to the shortest-path problem and solve it by using dynamic programming. Meng et al. [46] solve Bellman equations recursively, Rodrigues et al. [70] use integration techniques, and Arkian et al. [51] consider fuzzy logic and Q-learning.

4.6. Summary of Objectives in Resource Management. By far, the most active area of research in the edge resource management is resource allocation, as visible in Table 2. This is followed by optimization as a goal, where we see a great majority of papers present. Among the objectives from our taxonomy, resource estimation and resource discovery are least studied. Resource sharing, to the extent it is used, is well represented among the second and third types of architectures in Figure 1, that is, coordinator device and device clouds, but not in the first type of architecture (edge server).

Somewhat surprisingly, while scheduling is a major topic in cloud systems, the edge-specific literature does not consider it as the main problem, as evident from fewer works addressing scheduling compared to placement and migration. Where autoscaling is mentioned in an edge context, authors typically deal with offloading to the cloud, which was not the focus of our work. There are several excellent surveys already covering these. The work by Wang et al. [88] addressing autoscaling and the edge is among few exceptions, so we did not create a special category for this type of work.

While the previous breakdown was done in a resource independent manner, it is also interesting to consider the resource type studied with regard to the resource management objectives. Table 3 thus combines the information contained in Tables 1 and 2 to give us this view. Not surprisingly, most of the articles consider computation and communication for resource allocation and optimization. Quite expected as well, the proportion of resource sharing articles (from Table 2) considering energy as a resource (45% according to Table 3) is higher than the proportion of, for example, resource allocation articles considering energy (23%), as an incentive to share resources is when you consider energy-constrained devices. It is interesting to note that, in the surveyed works, resource estimation is most often done for a generic resource type and that none of the articles combined resource estimation and storage and resource discovery and data.

5. Resource Location

Computing at the edge differentiates itself from regular cloud computing with the fact that resources used can belong to different levels. It is indeed not uncommon to use resources at the edge level primarily but also from the cloud level if

Table 3: Surveyed articles according to resource type and objective of resource management.

		Objective				
		Resource estimation	Resource discovery	Resource allocation	Resource sharing	Resource optimization
Resource type	Computation	[20]	[51, 73, 80]	[20, 44, 46, 48–50, 53, 59, 61–63, 66–70, 72, 73, 75, 76, 78]	[20, 50–52, 64, 71, 74, 80]	[20, 46, 48–52, 59, 61–63, 66, 67, 70, 73, 76, 78, 80]
	Communication	[20]	[51, 73, 80]	[20, 44–46, 48–50, 53, 59–61, 63, 66–68, 70, 72, 73, 75, 76, 78, 79]	[20, 50–52, 55, 64, 74, 80, 81]	[20, 45, 46, 48–52, 55, 59, 61, 63, 66, 67, 70, 73, 76, 78, 80, 81]
	Storage		[51]	[45, 48, 50, 60, 61, 66, 75]	[50, 51]	[45, 48, 50, 51, 61, 66]
	Data	[20]		[13, 20, 50, 79]	[20, 50, 52, 71]	[20, 50, 52]
	Energy	[54, 83]	[54, 80]	[45, 62, 63, 65, 78, 82, 83]	[52, 74, 80, 81, 83]	[45, 52, 54, 62, 63, 78, 80–83]
	Generic	[43, 54, 58, 77]	[54]	[56, 77, 79, 82]	[55]	[54, 55, 77, 82]

required. Moreover, end devices and sometimes edge devices do not have to be stationary as in a data center. Note that here we make a distinction between mobility on the demand side and mobility on the supply side. Even though the demand side clients are almost always mobile, the infrastructure that supplies the adequate resources has been invariably stationary in the past.

In this section, we first look at where the *managed* resources considered are located within the architectures presented in Figure 1. We then shift focus and look at the same set of resources again but this time studying their mobility.

5.1. Location within the Architecture. Edge resource management is actually not only about managing resources located at the edge level as a study of the managed resources' location in the surveyed work reveals. This study is presented in Table 4.

5.1.1. Single Level. As expected when surveying edge resource management papers, a large part (54%) of those consider managed resources located only at the edge level, for example, the works by Arkian et al. [61], Fan et al. [62], Gomes et al. [13], Yousaf and Taleb [48], Chen and Xu [74], Sardellitti et al. [78], and Wang et al. [75].

Aazam et al. [43] consider resources located at only one physical location, a fog node, but considering resources within the same architectural level most often does not mean that the resources are located at the same physical location. For example, Oueis [63] considers resources on different cells and Gu et al. [66] and Plachy et al. [68] consider resources on different base stations. Fricker et al. [69] and Rodrigues et al. [70] consider task placement and migration on different types of edge devices (data centers for the former and cloudlets for the latter).

Essentially refining our architecture, some works distinguish different levels in the same architectural level from our Figure 1. For example, Wang et al. [49] consider transmission in the access network and computation in a mobile cloud computing architecture. Tärneberg et al. [67] consider that data centers at the edge can have a different distance to the device and different sizes.

Among the surveyed works, two works consider resources located only at the device level but where the management is performed at the edge, Tang et al. [64] and Nishio et al. [52], who consider resources present on different end devices.

There is no work considering managed resources located at the cloud level only as those were on purpose considered out of the scope of this survey.

5.1.2. Multilevel. We observe that resources do not need to belong to the same architecture level. Among the multilevel works, the most common is to use resources located both at the edge and at the cloud level. This is the case in the works by Liu et al. [59], Borylo et al. [65], Valancius et al. [45], Yi et al. [76], Wang et al. [77], and Singh et al. [44]. Specifically, Skarlat et al. [50] and Bittencourt et al. [72] favor using edge resources over cloud resources. Liu et al. [54] use resources in the device/edge level or in the cloud depending on the availability of the resources and Confais et al. [60] work with different storage locations at the edge or cloud level.

This is, however, not the only combination and Zhang et al. [71] work with data as a resource that can be located both in the end devices and at the edge. This combination is also used by Bianzino et al. [81] and Habak et al. [20], where an end device is promoted to an edge role.

Finally, combining the three levels, Zamani et al. [73] use resources on the device, on the network path to the cloud (edge level), and in the cloud level.

5.2. Resource Mobility. In an edge context, it is not obvious that resources located in the lower two levels of the architecture will be stationary or mobile. Therefore, it is interesting to study the mobility of the managed resources in the surveyed articles.

5.2.1. Stationary Resources. Most of the surveyed articles (71%) consider resources that are stationary only. This can be because the architecture/application considered does not have mobile resources or for simplification reasons. The latter is found in works where the architecture presented has

Table 4: Managed resources and their supply-side mobility.

	Article	Managed resources' location		
		Device level	Edge level	Cloud level
Edge server	Liu et al. [59]		Stationary	Stationary
	Confais et al. [60]		Stationary	Stationary
	Aazam et al. [43]		Stationary	
	Arkian et al. [61]		Stationary	
	Aazam and Hu [58]		Stationary + mobile	
	Fan et al. [62]		Stationary	
	Oueis [63]		Stationary	
	Tang et al. [64]	Stationary		
	Borylo et al. [65]		Stationary	Stationary
	Yousaf and Taleb [48]		Stationary	
	Wang et al. [49]		Stationary	
	Gu et al. [66]		Stationary	
	Tärneberg et al. [67]		Stationary	
	Plachy et al. [68]		Stationary	
	Gomes et al. [13]		Stationary	
	Fricker et al. [69]		Stationary	
	Rodrigues et al. [70]		Stationary	
	Zhang et al. [71]	Stationary	Stationary	
	Bittencourt et al. [72]		Stationary	Stationary
	Zamani et al. [73]	Stationary	Stationary	Stationary
	Valancius et al. [45]		Stationary	Stationary
	Chen and Xu [74]		Stationary	
	Wang et al. [75]		Stationary	
	Yi et al. [76]		Stationary	Stationary
	Wang et al. [77]		Stationary	Stationary
	Sardellitti et al. [78]		Stationary	
	Singh et al. [44]		Stationary	Stationary
Coordinator device	Nishio et al. [52]	Stationary		
	Skarlat et al. [50]		Stationary	Stationary
	Borgia et al. [79]		Mobile	Stationary
	Athwani and Vidyarthi [80]		Mobile	
	Arkian et al. [51]		Mobile	
	Penner et al. [56]		Mobile	
	Bianzino et al. [81]	Mobile	Mobile	
	Habak et al. [20]	Mobile	Mobile	
Device cloud	Liu et al. [54]		Stationary	Stationary
	Mascitti et al. [53]		Mobile	
	Liu et al. [55]		Mobile	
	Meng et al. [46]		Stationary + mobile	Stationary
	Qi et al. [82]		Mobile	Stationary
	Mtibaa et al. [83]		Mobile	

resources that are theoretically mobile but where this part is ignored in the solution or evaluation presented, for example, in [54] or [52].

This preponderance of stationary resources may be explained by the fact that those works consider edge as an extension of the cloud, which has only stationary resources.

5.2.2. Mobile Resources. Having mobile edge devices and thus mobile resources obviously creates lots of challenges such as how to handle the unreliable connectivity of those resources and how to provide seamless handovers. Thus, having mobile resources introduces another level of complexity in resource management algorithms.

Different mobility models are used; for example, Penner et al. [56] model departure and arrival times using statistical models, which is similar to what is used by Bianzino et al. [81]. Also using statistical models, Habak et al. [20] model arrival rate and presence time. In those statistical models, arrivals are modeled using a Poisson distribution, departure most often using an exponential distribution, and presence time using a normal distribution. Another model that is relatively common is the Random Way Point Model, used by Mascitti et al. [53] and Liu et al. [55].

With a different and more uncommon approach, Arkian et al. [51] consider the speed of the vehicles, and Athwani and Vidyarthi [80] consider that 10% of the nodes are moving after a request. Finally, Mtibaa et al. [83] consider both a mobility model with low disconnection rate and a mobility model based on a dataset (Infocom06), where the mobility of the devices is predictable in different communication scenarios.

5.2.3. Combination of Stationary and Mobile Resources. Some works mention a combination of mobile and stationary resources. In the edge level, Aazam and Huh [58] consider different types of devices (stationary or mobile). However, the devices are actually not mobile in their simulations.

Borgia et al. [79] consider the local cloud (i.e., the edge level) as mobile and the global cloud as stationary. They use the Random Way Point Model for mobility. Similarly, Qi et al. [82] have mobile end devices and stationary infrastructure servers and describe their own mobility model. Meng et al. [46] use a mobile vehicular cloud together with a stationary local cloud at the edge level and a stationary remote cloud. The mobility of the vehicles is modeled as a Poisson process.

5.3. Summary of Edge Resource Location. Table 4 reveals the distribution of the papers among the above categories and clearly shows that fewer works are multilevel, and the majority are stationary. As noted before, few works are studying managed resources located at different levels and/or mobile.

Note that this does not mean that the works do not consider mobility at all; it only means that the mobility is not on the supply side. An example of the works including mobility on the demand side only is the paper by Plachy et al. [68] who consider that computational resources needed by a user are allocated in a stationary base station in a VM, which can be transferred to another base station if the user is moving. Similar solutions are presented by Tärneberg et al. [67], Gomes et al. [13], Oueis [63], Fan et al. [62], and Wang et al. [77].

Despite demand side node mobility that may be present in all architectures, the supply side node mobility, that is, the notion of mobile managed resource, is among the promises of what the edge brings. We see more mobile resources present in the second and third types of architecture (coordinator device and device cloud). It remains to be seen if the future works will include more 3-level works in which at least two are mobile.

6. Resource Use

The final aspect of resource management considered in our taxonomy is the purpose for which the resource will be used.

6.1. Functional Properties. Edge computing is promoted as a means of getting access to a given service in most of the surveyed articles, that is, for satisfying functionality in an application. There are numerous articles in the literature providing an overview of edge applications, including [6, 7, 10, 23, 27, 34, 41, 89]. Such applications range over augmented reality, connected vehicles, disaster recovery, and a lot of others.

When looking at the different applications used in the surveyed articles presented in the earlier sections, the first finding is that the majority of them (66%) do not consider a specific application in their study. Instead, they refer to generic applications such as IoT services [60], real-time applications [68], and latency-sensitive applications [59] or name some applications but only as an illustration.

Table 5 presents the remaining papers according to which type of application they consider. We can distinguish seven areas in which the described applications can be categorized. Note that in the Generic category we place papers that although not fixed towards one domain of application refer specifically to classes of applications that they exemplify clearly.

6.2. Nonfunctional Properties. In addition to enabling functionalities when using the edge computing paradigm, the very organization of the edge architecture and realizing desirable properties require some kind of resource management too. This additional work is not directly related to the service to obtain; that is, it is a nonfunctional property (also referred to as extrafunctional properties). Obviously, papers that are focusing on a functional property can also be interested in some nonfunctional property.

This subsection is related to the categories of objectives for resource management we have already discussed in Section 4. Achieving the objectives in that section was evaluated using metrics that are often representative for measuring nonfunctional properties.

Examples of metrics and their related nonfunctional property which are encountered more often are

(i) *response time* as a measure of *timeliness*

(ii) *energy consumption* as a measure of *energy efficiency*

(iii) *admission ratio*, or its equivalent blocking probability, as a measure of *availability* of the edge service

(iv) *CPU/network utilization* as a measure of *computation/communication resource efficiency*

(v) *monetary cost* paid to an infrastructure owner as a measure of *cost efficiency*

The list of metrics is not exhaustive, but we have focused on the more prevalent ones. Figure 4 shows how popular the above metrics are in the context of the works studied so far.

Table 5: Applications considered in the surveyed articles.

Area	Applications	Articles
Healthcare	Medical cyber-physical systems	[66]
	Connected health	[71]
Video	Video analytics	[73, 76]
	Video surveillance	[71, 72, 75]
	Video on demand	[45]
IoT	Crowd-sensing	[61]
	Sense-Process-Actuate application	[50]
Gaming	Electroencephalography (EEG) tractor beam game	[72]
Transportation	Connected vehicles	[46, 51]
Content management	User profiling	[13]
Generic	Computation/communication-intensive	[70]
	Delay-sensitive/Delay-tolerant	[81]

Figure 4: Generic metrics related to a nonfunctional property used in the surveyed articles.

It is not surprising that those metrics that relate to timeliness or availability or resource efficiency are well represented.

As we have noticed earlier, the same paper can deal with multiple resources, multiple objectives, and, also clearly seen in this figure, multiple nonfunctional properties. This illustrates the complex trade-offs involved when dealing with resources in a multistakeholder distributed system.

7. Research Challenges

In this section, we present the research challenges not substantially addressed which could be of interest for further research in the field.

From the previous sections, we noted that the architecture with three active and distinct levels (edge server) is predominant. We also noted that the resource objectives allocation and optimization were well studied. Moreover, computational and communicational resources are the most commonly addressed, typically being stationary and located within a single level. Therefore, research is less prevalent on data, storage, and energy as a resource and less extensive towards the estimation, discovery, and sharing objectives (especially the first two). Furthermore, new works should consider mobility and multilevel locality on the supply side.

Elaborating on mobility, the new phenomenon at the edge is that the supply side can also be mobile and not only the demand side as it was the case in classic clouds. Indeed, edge systems will have to deal with a greater variety of mobility with end devices that are often mobile (like vehicles) but can also be stationary (e.g., video-surveillance

cameras), as well as mobile edge devices. It is, however, not obvious that the mobility patterns of all those devices will be similar, especially between end and edge devices. Considering the large variety of edge applications, their characteristics can potentially vary greatly. For example, an edge solution intended to serve networks of cars moving on a road network will probably be quite different from an edge solution intended to serve persons within a shopping mall. Hence, it is critical to have efficient and thus tailored solutions. But should each application domain rediscover the wheel? Obviously, there is going to be generic wisdom that is transferable across the domains if adequate characterizations of resource requirement patterns are formulated. More work is needed on collecting mobility traces from the different edge applications to see if present patterns can in a generic way be used to create pertinent edge mobility models at both levels of the architecture, the end and the edge level. These can then become a basis for repeatable evaluations of resource management strategies.

Another aspect that will be critical to solving is collaboration. There are new papers appearing where multiple operators at the edge level are modeled, and this introduces new challenges. At the end level, we have seen that different incentives can be provided to enable resource sharing [64, 81] and similarly at the edge level [74]. Such collaboration is especially good for managing workload churns and is interesting for infrastructure owners. The next challenge would be to do multilevel collaboration with a hierarchy of incentive schemes at different levels assuring that they do not cancel out each other's benefits. Moreover, finding more advanced incentive schemes that take both resource efficiency and security into account is needed. Current solutions either choose not to collaborate for security- or privacy-sensitive tasks [44] or rely on classic trust establishment [74] but this will not be enough for a wide collaboration at the edge.

Context adaptation is also one of the properties expected from edge computing and is advocated as a good reason to choose this paradigm [90]. Providing tailored service depending on the user's physical location of course has to be taken care of at the application level. However, it also impacts resource management as those applications will require resources to provide those services, in particular considering data (about supply mobility and abundance) as a resource.

Security, and its subcomponents availability, confidentiality, and integrity, is a key point for edge computing, together with privacy with respect to sensitive end-user data. Although similar, security and privacy have distinct characteristics and should be addressed in depth and separately, which is not the case in the current surveyed works [44, 74]. Regarding availability, most of the works considered focus on admission ratio but do not consider the fact that resources could disappear while executing due to mobility, misuses, or attacks. A notable exception is the work by Habak et al. [20] who propose and evaluate a task checkpointing mechanism that performs result replication to mitigate in case a device disappears. Focusing on availability, several works always consider that the cloud is available as a last fall-back for providing an edge service. If this is not the case (e.g., due to overload, attack, or natural disaster), the availability of the edge service will be impacted. More works in those directions and quantifying edge-specific availability metrics are required. Edge computing will most certainly be interesting for critical infrastructure because of its benefits and those require high standards on security. Research in this direction can be found for the mobile cloud paradigm, for example [91, 92], but they consider scenarios where the edge level is absent.

End-to-end timeliness requires quantification of latencies from an end device towards the cloud (or somewhere at the edge) all the way back to the same device (or to another device). This means traversing the edge networking services, including what we referred to as resource management services in this paper. Since estimation, discovery, sharing, and allocation (including migration) are complex algorithms in such networks, these must also be evaluated in terms of their own resource footprint and thereby their own impact on timeliness and QoS. In the surveyed articles, computing time of the solution is only evaluated by Gomes et al. [13] and Skarlat et al. [50]. Since edge computing cannot become widely used without strong security and privacy properties, it is especially important to research on the resource overhead for providing those properties as well. Too high an overhead can signify a technology that is not feasible in practice.

As shown in Section 3, resources managed at the edge are most often a combination of different resource types. This implies that there will be some interrelations among resource utilization levels, which can create new challenges. Considering resource affinity as in the work by Yousaf and Taleb [48] may be a start but more research is needed to understand and address the complexity of such multiresource problems in the edge context.

As mentioned in Section 2, edge computing brings together diverse business sectors with their existing techniques for solving relevant problems in those areas. Techniques previously applied in only one of those domains may be applicable to edge computing with the required adaptations. For example, performing resource migration requires efficient techniques for this purpose. Ma et al. [93] study container migration and found that the hand-off time decreased by 56% to 80% in comparison to state-of-the-art VM migration for the edge. Results like this should be exploited in the new edge era and utilize technologies that may bring added benefit to edge computing.

Another enabler for resource-efficient edge computing is the development of tools for testing the new proposals in relevant conditions and setups. In the surveyed articles, the most common method used for validating a model or a proposed algorithm is to use an analytical tool (e.g., a solver and/or an optimization engine). Another common approach is to use a simulator, either a generic network simulator such as OMNeT++ (https://omnetpp.org/) or one designed for regular cloud environments such as CloudSim [94], most often with some custom extensions. There also exists a dedicated simulator designed for fog computing, called iFogSim [95], which extends CloudSim, but this one currently has limitations, for example, no mechanisms for offloading or communication between two nodes at the same

level. A third way of evaluating in the surveyed works is the use of physical testbeds. Such evaluations provide invaluable insights into problems that are easy to oversee in simulation and investigate their impact. However, a big challenge for testbeds is to get them to scale, which is to some extent also a problem for simulations. Therefore, there is a need for creating open research testbeds and simulation tools so that configurable architectures and application/domain-specific edge computing methods can be efficiently compared. Coming back to a previous point, such tools should be able to handle mobility of end and edge devices and should obviously be scalable for evaluation of real-world scenarios.

8. Conclusion

The past decade has created tremendous expectations on IoT changing the landscape of data-driven services with benefits for multiple societal sectors. Many researchers have contributed to the development of technologies and addressed challenges that come with resource scarcity in the end devices. Other researchers with a background in cloud computing have looked at how to carry the data generated by the massive IoT deployments and how to efficiently use the cloud resources. The area of edge computing brings these two ends of the same service together in an emerging ecosystem and creates a means to discuss resource adequacy from an end-to-end perspective. In this paper, we have tried to provide an overview, not from a cloud perspective or an IoT device perspective, but with a focus on edge resource management.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Swedish National Graduate School in Computer Science (CUGS).

References

[1] Ericsson, *Ericsson Mobility Report*, 2017, https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.

[2] M. Satyanarayanan, P. Simoens, Y. Xiao et al., "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

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

[4] A. Mehta, W. Tärneberg, C. Klein, J. Tordsson, M. Kihl, and E. Elmroth, "How Beneficial Are Intermediate Layer Data Centers in Mobile Edge Networks?" in *Proceedings of the IEEE Workshops on Foundations and Applications of Self∗ Systems (FAS∗W)*, pp. 222–229, Augsburg, Germany, September 2016.

[5] M. Etemad, M. Aazam, and M. St-Hilaire, "Using DEVS for modeling and simulating a Fog Computing environment," in *Proceedings of the 2017 International Conference on Computing, Networking and Communications, (ICNC)*, pp. 849–854, Santa Clara, Calif, USA, January 2017.

[6] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the 1st ACM Mobile Cloud Computing Workshop, MCC '12*, pp. 13–16, ACM, Helsinki, Finland, August 2012.

[8] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, no. Part 2, pp. 680–698, 2018.

[9] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara, "Cloudpath: a multi-tier cloud computing framework," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, vol. 13, pp. 1–20, ACM, San Jose, Calif, USA, October 2017.

[10] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile Edge Cloud System: Architectures, Challenges, and Approaches," *IEEE Systems Journal*, pp. 1–14, 2017.

[11] W. Tärneberg, A. Mehta, J. Tordsson, M. Kihl, E. Elmroth, and W. Tärneberg, "Resource management challenges for the infinite cloud," in *Proceedings of the 10th International Workshop on Feedback Computing at CPSWeek*, Lund University, Seattle, Wash, USA, 2015.

[12] T. Taleb and A. Ksentini, "Follow Me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[13] A. S. Gomes, B. Sousa, D. Palma et al., "Edge caching with mobility prediction in virtualized LTE mobile networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.

[14] V. Xhagjika, L. Navarro, and V. Vlassov, "Enhancing real-time applications by means of multi-tier cloud federations," in *Proceedings of the 7th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 397–404, IEEE, Vancouver, Canada, December 2015.

[15] F. Lobillo, Z. Becvar, M. A. Puente et al., "An architecture for mobile computation offloading on cloud-enabled LTE small cells," in *Proceedings of the 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, IEEE, Istanbul, Turkey, April 2014.

[16] K. Wang, A. Banerjee, M. Shen, J. Van der Merwe, J. Cho, and K. Webb, "MobiScud: A fast moving personal cloud in the mobile network," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pp. 19–24, ACM, London, UK, 2015.

[17] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "Concert: A cloud-based architecture for next-generation cellular systems," *IEEE Wireless Communications Magazine*, vol. 21, no. 6, article no. A9, pp. 14–22, 2014.

[18] P. T. Endo, A. V. De Almeida Palhares, N. N. Pereira et al., "Resource allocation for distributed cloud: Concepts and research challenges," *IEEE Network*, vol. 25, no. 4, pp. 42–46, 2011.

[19] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: leveraging mobile devices to provide cloud service at the edge," in *Proceedings of the 8th IEEE International Conference on Cloud Computing*, pp. 9–16, IEEE, New York, NY, USA, July 2015.

[20] K. Habak, E. W. Zegura, M. Ammar, and K. A. Harras, "Workload management for dynamic mobile device clusters

in edge femtoclouds," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–14, ACM, San Jose, Calif, USA, October 2017.

[21] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proceedings of the 10th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–8, IEEE, Coimbatore, India, January 2016.

[22] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proceedings of the 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pp. 117–122, Southbank, Australia, November 2014.

[23] M. Chiang and T. Zhang, "Fog and IoT: an overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[25] N. M. Gonzalez, W. A. Goya, R. de Fatima Pereira et al., "Fog computing: Data analytics and cloud distributed processing on the network edges," in *Proceedings of the 2016 35th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–9, Valparaíso, Chile, October 2016.

[26] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: a survey," in *Proceedings of the 10th International Conference on Wireless Algorithms, Systems, and Applications*, vol. 9204 of *Lecture Notes in Computer Science*, pp. 685–695, Springer International Publishing, 2015.

[27] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit From Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[28] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the Workshop on Mobile Big Data (Mobidata '15)*, pp. 37–42, ACM, Hangzhou, China, June 2015.

[29] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions," in *Internet of Everything. Internet of Things (Technology, Communications and Computing)*, pp. 103–130, Springer, Singapore, 2018.

[30] A. Bhattacharya and P. De, "A survey of adaptation techniques in computation offloading," *Journal of Network and Computer Applications*, vol. 78, pp. 97–115, 2017.

[31] F. Rebecchi, M. Dias de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 580–603, 2015.

[32] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile Edge Computing: Survey and Research Outlook," https://arxiv.org/abs/1701.01090, 2017.

[33] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[34] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, 2017.

[35] L. Chunlin and L. LaYuan, "Cost and energy aware service provisioning for mobile client in cloud computing environment," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1196–1223, 2015.

[36] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, "INDICES: exploiting edge resources for performance-aware cloud-hosted services," in *Proceedings of the 1st IEEE International Conference on Fog and Edge Computing (ICFEC)*, pp. 75–80, IEEE, Madrid, Spain, 2017.

[37] K. Toczé and S. Nadjm-Tehrani, "Where Resources Meet at the Edge," in *Proceedings of the 2017 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 302–307, IEEE, Helsinki, Finland, August 2017.

[38] R. Cziva and D. P. Pezaros, "Container Network Functions: Bringing NFV to the Network Edge," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 24–31, 2017.

[39] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges," *IEEE Communications Surveys & Tutorials*, 2017.

[40] A. Mtibaa, K. A. Harras, K. Habak, M. Ammar, and E. W. Zegura, "Towards mobile opportunistic computing," in *Proceedings of the 8th IEEE International Conference on Cloud Computing*, pp. 1111–1114, IEEE, New York, NY, USA, July 2015.

[41] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[42] H. Frank, W. Fuhrmann, and B. Ghita, "Mobile Edge Computing: Requirements for powerful mobile near real-time applications," in *Proceedings of the 11th International Network Conference (INC 2016)*, pp. 63–66, Germany, July 2016.

[43] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "PRE-Fog: IoT trace based probabilistic resource estimation at Fog," in *Proceedings of the 13th IEEE Annual Consumer Communications and Networking Conference (CCNC 2016)*, pp. 12–17, IEEE, Las Vegas, Nev, USA, January 2016.

[44] A. Singh, N. Auluck, O. Rana, A. Jones, and S. Nepal, "RT-SANE: Real time security aware scheduling on the network edge," in *Proceedings of the10th International Conference on Utility and Cloud Computing*, pp. 131–140, ACM, New York, NY, USA, 2017.

[45] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pp. 37–48, ACM, Rome, Italy, December 2009.

[46] H. Meng, K. Zheng, P. Chatzimisios, H. Zhao, and L. Ma, "A utility-based resource allocation scheme in cloud-assisted vehicular network architecture," in *Proceedings of the IEEE International Conference on Communication Workshop, ICCW 2015*, pp. 1833–1838, IEEE, London, UK, June 2015.

[47] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[48] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5G carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, 2016.

[49] K. Wang, K. Yang, X. Wang, and C. S. Magurawalage, "Cost-effective resource allocation in C-RAN with mobile cloud," in *Proceedings of the IEEE International Conference on Communications, ICC 2016*, IEEE, Kuala Lumpur, Malaysia, May 2016.

[50] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-Aware Fog Service Placement," in *Proceedings of the 1st IEEE International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 89–96, IEEE, Madrid, Spain, 2017.

[51] H. R. Arkian, R. E. Atani, and A. Pourkhalili, "A cluster-based vehicular cloud architecture with learning-based resource management," in *Proceedings of the 2014 6th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2014*, pp. 162–167, IEEE, Singapore, December 2014.

[52] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *Proceedings of the First International Workshop on Mobile Cloud Computing & Networking*, pp. 19–26, ACM, New York, NY, USA, 2013.

[53] D. Mascitti, M. Conti, A. Passarella, and L. Ricci, "Service Provisioning through Opportunistic Computing in Mobile Clouds," *Procedia Computer Science*, vol. 40, pp. 143–150, 2014.

[54] W. Liu, T. Nishio, R. Shinkuma, and T. Takahashi, "Adaptive resource discovery in mobile cloud computing," *Computer Communications*, vol. 50, pp. 119–129, 2014.

[55] W. Liu, R. Shinkuma, and T. Takahashi, "Opportunistic resource sharing in mobile cloud computing: The single-copy case," in *Proceedings of the 16th Asia-Pacific Network Operations and Management Symposium, APNOMS 2014*, pp. 1–6, IEEE, Hsinchu, Taiwan, September 2014.

[56] T. Penner, A. Johnson, B. Van Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *Proceedings of the 2014 IEEE Global Communications Conference, GLOBECOM 2014*, pp. 2801–2806, IEEE, Austin, Tex, USA, December 2014.

[57] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and Software Architecture for Fog Computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, 2017.

[58] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications (AINA '15)*, pp. 687–694, IEEE, Gwangiu, South Korea, March 2015.

[59] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, 2016.

[60] B. Confais, A. Lebre, and B. Parrein, "An Object Store Service for a Fog/Edge Computing Infrastructure Based on IPFS and a Scale-Out NAS," in *Proceedings of the 1st IEEE International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 41–50, IEEE, Madrid, Spain, 2017.

[61] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[62] Q. Fan, N. Ansari, and X. Sun, "Energy Driven Avatar Migration in Green Cloudlet Networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, 2017.

[63] J. Oueis, *Joint Communication and Computation Resources Allocation for Cloud-Empowered Future Wireless Networks [Ph.D. thesis]*, Université Grenoble, 2016.

[64] L. Tang, S. He, and Q. Li, "Double-Sided Bidding Mechanism for Resource Sharing in Mobile Cloud," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1798–1809, 2017.

[65] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk, "Energy-aware fog and cloud interplay supported by wide area software defined networking," in *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, Kuala Lumpur, Malaysia, May 2016.

[66] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2017.

[67] W. Tärneberg, A. Mehta, E. Wadbro et al., "Dynamic application placement in the Mobile Cloud Network," *Future Generation Computer Systems*, vol. 70, pp. 163–177, 2017.

[68] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proceedings of the 27th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, IEEE, Valencia, Spain, September 2016.

[69] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an Offloading Scheme for Data Centers in the Framework of Fog Computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 4, pp. 1–18, 2016.

[70] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[71] Q. Zhang, X. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Big data sharing and processing in collaborative edge environment," in *Proceedings of the 4th IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2016*, pp. 20–25, Washington, DC, USA, October 2016.

[72] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-Aware Application Scheduling in Fog Computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[73] A. R. Zamani, M. Zou, J. Diaz-Montes et al., "Deadline Constrained Video Analysis via In-Transit Computational Environments," *IEEE Transactions on Services Computing*, pp. 1-1, 2017.

[74] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proceedings of the the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–11, ACM, San Jose, Calif, USA, October 2017.

[75] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proceedings of the the Workshop on Smart Internet of Things*, pp. 1–6, ACM, San Jose, Calif, USA, October 2017.

[76] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," in *Proceedings of the the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–13, ACM, San Jose, Calif, USA, October 2017.

[77] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017.

[78] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[79] E. Borgia, R. Bruno, M. Conti, D. Mascitti, and A. Passarella, "Mobile edge clouds for Information-Centric IoT services," in *Proceedings of the 2016 IEEE Symposium on Computers and Communication, ISCC 2016*, pp. 422–428, IEEE, Messina, Italy, July 2016.

[80] P. Athwani and D. P. Vidyarthi, "Resource discovery in mobile cloud computing: A clustering based approach," in *Proceedings*

of the IEEE UP Section Conference on Electrical Computer and Electronics, UPCON 2015, IEEE, Allahabad, India, December 2015.

[81] A. P. Bianzino, J. Rougier, C. Chaudet, and D. Rossi, "The Green-Game: Accounting for Device Criticality in Resource Consolidation for Backbone IP Networks," *Strategic Behavior and the Environment*, vol. 4, no. 2, pp. 131–153, 2014.

[82] Q. Qi, J. Liao, J. Wang, Q. Li, and Y. Cao, "Dynamic resource orchestration for multi-task application in heterogeneous mobile cloud computing," in *Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 221–226, IEEE, San Francisco, Calif, USA, April 2016.

[83] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar, "Towards resource sharing in mobile device clouds: Power balancing across mobile devices," in *Proceedings of the 2013 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC 2013*, pp. 51–56, ACM, Hong Kong, China, August 2013.

[84] E. J. Vergara, S. Nadjm-Tehrani, and M. Asplund, "Fairness and Incentive Considerations in Energy Apportionment Policies," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 2, no. 1, pp. 1–29, 2016.

[85] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[86] Z. Dong, Y. Liu, H. Zhou et al., "An energy-efficient offloading framework with predictable temporal correctness," in *Proceedings of the the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–12, ACM, San Jose, Calif, USA, October 2017.

[87] R. Yu, X. Huang, J. Kang et al., "Cooperative resource management in cloud-enabled vehicular networks," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7938–7951, 2015.

[88] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: a framework for edge node resource management," *IEEE Transactions on Services Computing*, 2017.

[89] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49, 2013.

[90] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service," in *Proceedings of the 8th IEEE International Conference on Cloud Computing*, pp. 869–876, IEEE, New York, NY, USA, July 2015.

[91] Y. Liu and M. J. Lee, "Security-Aware Resource Allocation for Mobile Cloud Computing Systems," in *Proceedings of the 2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, IEEE, Las Vegas, Nev, USA, August 2015.

[92] H. Liang, D. Huang, L. X. Cai, X. Shen, and D. Peng, "Resource allocation for security services in mobile cloud computing," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, pp. 191–195, IEEE, Shangha, China, April 2011.

[93] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proceedings of the the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–13, ACM, San Jose, Calif, USA, October 2017.

[94] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[95] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 4593208, 11 pages
https://doi.org/10.1155/2018/4593208

Research Article

AIMING: Resource Allocation with Latency Awareness for Federated-Cloud Applications

Jie Wei, Ao Zhou, Jie Yuan, and Fangchun Yang

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Haidian, Beijing 100876, China

Correspondence should be addressed to Jie Wei; wjwb@bupt.edu.cn

Received 27 October 2017; Revised 24 January 2018; Accepted 14 February 2018; Published 10 April 2018

Academic Editor: Anna Kobusinska

Federated-cloud has been widely deployed due to the growing popularity of real-time applications, and hence allocating resources among clouds becomes nontrivial to meet the stringent service requirements. The challenges lie in achieving minimized latency constrained by virtual machines rental overhead and resource requirement. This becomes further complicated by the issues of datacenter selection. To this end, we propose AIMING, a novel resource allocation approach which aims to minimize the latency constrained by monetary overhead in the context of federated-cloud. Specifically, the network resources are deployed and selected according to k-means clustering. Meanwhile, the total latency among datacenters is optimized based on binary quadratic programming. The evaluation is conducted with real data traces. The results show that AIMING can reduce total datacenter latency effectively compared with other approaches.

1. Introduction

Real-time applications continue to grow rapidly, with evermore functionality and evermore users around the globe. Because of this growth, major real-time application providers now use tens of geographically dispersed datacenters to support service. For early real-time application providers, they have an urgent need to migrate their service from their own servers to distributed datacenters to ensure short latency and compete with new application providers. Hence, the main unmet challenge of leveraging these datacenters is effectively allocating resource and optimizing latency for applications. Usually the resource comes in the form of virtual machines (VM). The application providers deploy different types of VM configurations which consist of specific amount of CPU, memory, and disk resource.

It is challenging to determine the location and the number of VMs for application providers, without any knowledge of the latency among datacenters (since the application providers are in the SaaS layer). However, the cloud providers have their own fiber links to interconnect all major regions, and the latency among datacenters is stable and measurable. By this feature, it is feasible to determine the number of VMs and optimize delivery latency for application providers.

In this paper, a resource allocation approach AIMING (k-meAns & bInary quadratic programMING) is proposed for VM placement to optimize latency among datacenters for real-time application providers in federated-cloud. (The federated-cloud is the deployment and management of multiple external and internal cloud computing services to meet business needs.) The AIMING optimizes total latency from the perspective of application providers and provides a VMs rental strategy for real-time application providers, especially for the early providers who need to migrate their services to federated-cloud. We get the measured latency among datacenters and then select datacenters according to the latency. There are several constraints to be addressed.

(1) Choosing the well-connected datacenters with low latency is essential, since there are dozens of datacenters around the world and the paths' latency among these datacenters is different. Latency significantly affects the QoE of real-time application users. Thus, ensuring low latency is the initial aim of the application providers.

(2) Compared to standalone datacenter, different datacenters have different VM rental prices in federated-cloud.

An application provider may choose a cheaper datacenter even if with higher latency. The application providers have their own budget for renting VMs. Hence, latency optimization should be constrained by the limitation of monetary overhead.

(3) In addition to the VMs rental overhead, the network overhead also should be considered. Real-time applications such as VoIP may generate accumulative huge data along with the increasing duration time since the data interaction exists among VMs located in different datacenters.

Many latency optimization approaches have been proposed for latency-sensitive applications in federated-cloud scenario. Hao et al. [1] have proposed an online algorithm to guarantee the service latency. Guo et al. [2] and Grozev and Buyya [3] optimize latency for virtual desktops and web applications, respectively, by calculating. Aral and Ovatman [4] have optimized the latency for interdatacenters and intradatacenters by heuristic algorithm. Alicherry and Lakshman [5] have proposed an algorithm to minimize network latency and bandwidth utilization based on mapping VM clusters onto the federated-cloud. Compared with existing resource allocation approaches, the main contributions of our work are listed as follows:

(1) We propose the AIMING to allocate resource (VMs and network) from the perspective of application providers with different types of VMs taken into consideration.

(2) The distribution of federated-cloud is not uniform. We use k-means to classify the datacenters into several regions according to real-world latency data traces, which is better than functional calculation method. The classification is more accurate and effective than graph theory since we use within-cluster sum of squares (WCSS) to evaluate.

(3) The monetary overhead and latency are a couple of tradeoffs. The application providers may limit budget for each region of datacenters. Small budget may lead to higher latency. We optimize the latency under the constraint of budget.

This paper is aimed at advancing the current state-of-the-art researches in latency optimization of real-time applications. The proposed AIMING is two-fold: First, k-means is used to classify the datacenters according to latency. Then, the binary quadratic programming is utilized to place VMs for application providers in federated-cloud.

The rest of this paper is organized as follows. Section 2 introduces the related work. In Section 3, we use k-means to classify the datacenters. Then, we use binary quadratic programming to place VMs in Section 4. Section 5 presents the implementation, experiment, and analysis of the proposed approach, and we conclude the paper in Section 6.

2. Related Work

A number of schemes have been proposed for latency and monetary overhead optimization in cloud. In this section, we briefly review previous works related to this work from three aspects, namely, standalone cloud resource allocation, federated-cloud resource allocation, and mobile cloud resource allocation.

2.1. Standalone Cloud Resource Allocation. Many works allocate VMs in standalone cloud to reduce latency and monetary overhead. Meng et al. [6] have proposed a two-tier approximation algorithm to efficiently place VMs. The algorithm consists of two components: SlotClustering and VMMinKcut. SlotClustering is to partition n slots into k clusters by using the cost between slots as the partition criterion. VMMinKcut is to partition n VMs into k VM clusters with minimum intercluster traffic. Fang et al. [7] have designed VMPlanner to optimize both VM placement and traffic flow routing so as to turn off unneeded network elements for power saving. An offline VM placement through emulated VM migration is proposed by Li et al. [8]. As long as the suitable physical machine has enough capacity, the migration algorithm can place the VM to its best physical machine directly. Furthermore, they study a hybrid scheme by employing a batch to accept upcoming VMs for online scenarios. Ant colony is deployed by Gao et al. for resource allocation in cloud. Both of them optimize the power consumption of CPU utilization.

In addition, Gao et al. [9, 10] also take the potential cost of wasted resources into consideration. A cloud resource allocation based on imperfect information Stackelberg game (IISG) and hidden Markov model (HMM) has been proposed by Wei et al. [11]. They firstly use the HMM to predict the service provider's current bid based on demand. Then they establish the IISG to choose the optimal bidding strategy and achieve maximum profits. Li et al. [12] focus on cost optimization of network traffic and physical machines utilization. They solve the VM placement problem from two aspects. The first aspect is putting emphasis on the cost of network traffic with fixed physical machines cost for two cases (the number of requested VMs is same or different). The second aspect is placing VMs in general case and taking both network cost and physical machine utilization cost into consideration. Bandwidth allocation is considered by Tian et al. and Yu et al. A distributed host-based bandwidth allocation design with underlay congestion control layer and private congestion control layer has been presented by Tian et al. [13]. To address the issue that existing works do not consider the impact of primary embedding on backup resource consumption, Yu et al. [14] have proposed a novel algorithm to compute the most efficient resource embedding given a tenant request.

2.2. Federated-Cloud Resource Allocation. Some notable schemes are proposed for federated-cloud resource allocation. Jiao et al. and Wu and Madhyastha optimize the network traffic by resource allocation in federated-cloud. Multiobjective optimization for placing users' data over multiple clouds for socially aware services has been studied by Jiao et al. [15]. They build a framework that can accommodate different objectives. An optimization approach based on graph cuts is leveraged for decomposing their original problem. To minimize user perceived latency, Wu and Madhyastha [16] have conducted a deep study of latency benefits when deploying web-services across AWS, Microsoft Azure, and Google Compute Engine. Their measurement shows that it is necessary to replicate data to ensure low latency for users. The capacity of cloud infrastructure has been studied by Narayanan et al. and Tordsson et al. Towards a leaner

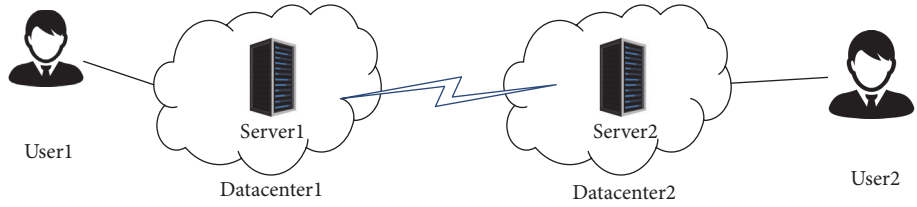

Figure 1: Distributed clouds communication network model for real-time application users.

geodistributed cloud infrastructure, Narayanan et al. [17] discuss several factors that influence the capacity provisioning and illustrate the research challenges for software design.

A novel cloud brokering approach has been proposed by Tordsson et al. [18] to optimize the placement of virtual infrastructure across multiple clouds. In a high throughput computing cluster case, they verify the feasibility of the approach. The communication overhead is considered by Lee et al. and Yi et al. Lee et al. [19] have proposed a distributed resource allocation approach for resource competition in federated-cloud. In their approach, each job consists of tasks and the communication behavior among tasks can be profiled. According to the communication behavior, their approach minimizes the communication overhead and tries to allocate grouped tasks to get balance in the case of resource competition. From the perspective of customers, Yi et al. [20] focus on network-aware resource allocation and develop a mixed binary quadratic programming optimal model to minimize the rental cost for each customer. Except for energy, the bandwidth is also considered by Xu and Li [21].

To solve the large-scale optimization, a distributed algorithm based on alternating direct method of multipliers has been proposed. To achieve energy efficiency and satisfy deadline constraints, Ding et al. [22] focus on dynamic scheduling of VMs. Mihailescu et al. and Wang et al. present pricing schemes for users according to their resource need. Mihailescu and Teo [23] have discussed a strategic-proof dynamic pricing scheme suitable for allocating resources in federated-cloud. A new cloud brokerage service is proposed by Wang et al. [24] based on leveraging both pricing benefits and multiplexing gains. The proposed service reserves a large pool of instances and serves users with price discounts. Hung et al. [25] coordinate job scheduling across datacenters with low overhead, while achieving near-optimal performance. A data hosting scheme has been proposed by Zhang et al. [26] to select several clouds to store data with monetary minimizing. Ardagna et al. [27] use game-theory to management runtime of resources from multiple IaaS providers to multiple SaaS with a revenue and penalty cost model. Jiao et al. [28] allocate and reconfigure resource in the multitier resource pool. And Palmieri et al. [29] optimize the overall communication and runtime resource utilization of cloud infrastructures by reoptimizing the communication paths between VMs and big data sources.

2.3. Mobile Cloud Resource Allocation. The resource allocation problem is also a big challenge for mobile cloud environment. Gai et al. [30] have proposed a dynamic energy-aware cloudlet-based mobile cloud computing model concentrating on energy consumption during the wireless communications. In their work, they have solved energy problem with dynamic network and provided guidelines and theoretical supports. To efficiently handle the peak load and satisfy the requirements of remote program execution, Tong et al. [31] have deployed cloud servers at the network edge and designed the edge cloud as a tree hierarchy of geodistributed servers. Furthermore, a workload placement algorithm is proposed to decide which edge cloud servers mobile programs should be placed on. There are also many works for mobile service optimization [32–34].

Different from the previous research, AIMING allocates resources by latency optimization with monetary overhead constraint. The resources contain CPU and memory. We combine the k-means and binary quadratic programming to solve the optimization problem.

3. Preliminary

To facilitate presenting the proposed approach, we first define some terms and notations that will be used for the paper in Notations. Then, we illustrate latency model and monetary overhead model for AIMING.

3.1. Latency Model. In this section, we introduce our latency model in detail. We first discuss the communication model of real-time application among two users in federated-cloud through datacenters. Based on this communication model, we then model the latency of data transfer among VMs.

We assume that a real-time application is distributed on federated-cloud and provides service for users through the datacenter network. As shown in Figure 1, user1 and user2 are users of real-time application and server1 and server2 are distributed in datacenter1 and datacenter2, respectively. Server1 in datacenter1 is the nearest available application server to user1, and it is the same as server2 and datacenter2. The message source user1 connects with destination user2 through datacenter1 and datacenter2.

In the federated-cloud scenario, each VM may contact with each other and transfer data. We first show the latency model between two VMs for data transmission. Let VM_p^{ij} denote pth VM of ith instance type in jth datacenter. And it is the same as VM_q^{lk}. $l(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk})$ is the latency between VM_p^{ij} and VM_q^{lk}. $v(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk})$ is the transmission data volume between VM_p^{ij} and VM_q^{lk}. We get the latency by ping. $t_{jk}(\mathrm{ping})$ is the time of ping, and $v(\mathrm{packet})$ is the packet

size of ping. Through t_{jk}(ping) and v(packet), we can model the latency between VM_p^{ij} and VM_q^{lk} as follows:

$$l\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) = v\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot \frac{t_{jk}\left(\mathrm{ping}\right)}{v\left(\mathrm{packet}\right)} \cdot x_p^{ij} \cdot x_q^{lk}, \tag{1}$$

where $x_p^{ij} \in \{0, 1\}$ and $x_p^{ij} = 1$ indicates that VM_p^{ij} is selected to rent for service; otherwise $x_p^{ij} = 0$. And it is the same as x_q^{lk}. The latency between VMs approximately equals the latency between datacenters where VMs are located, since the datacenter is extremely large. The formula of total latency can be shown as follows:

$$l_{\mathrm{total}} = \sum_{j,k=1}^{n_{\mathrm{dc}}} \sum_{i,l=1}^{n_{\mathrm{type}}} \sum_{p,q=1}^{n(\mathrm{VM}^{ij})} l\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right). \tag{2}$$

n_{dc} is the number of datacenters where VMs are placed. n_{type} is the number of the types of VM instances. $n(\mathrm{VM}^{ij})$ is the VM number of ith instance types in jth datacenter. We consider the data communication among each VM in formula (2). After finishing the latency model, we show the monetary overhead model in the following Section 3.2.

3.2. Monetary Overhead Model. We depict the monetary overhead model in this section. First, we discuss the data communication mode among VMs and pricing mechanism of federated-cloud. Figure 2 shows the data communication model between two VMs. VM 1 and 2 have different rental prices, and the data communication also needs expense. Thus, the total monetary overhead consists of two parts: VM rental monetary overhead and data transmission monetary overhead.

In addition, for federated-cloud scenario, different cloud providers have different prices. Even if the datacenters belong to the same cloud provider, they may have different VM rental prices and data transmission prices. The pricing mechanism of federated-cloud is shown in Figure 3. There are three cloud providers A, B, and C, and each provider has two datacenters. Users can get the prices of datacenters from the pricing module. They can choose the datacenters to rent VMs according to the pricing module.

According to the above discussion, the total monetary overhead consists of two parts: VM rental monetary overhead and data transmission monetary overhead. Different datacenters have different VM rental prices and data transmission prices. We model the VM rental monetary overhead and data transmission monetary overhead separately as follows:

$$m_{\mathrm{rental}} = \sum_{j=1}^{n_{\mathrm{dc}}} \sum_{i=1}^{n_{\mathrm{type}}} \sum_{p=1}^{n(\mathrm{VM}^{ij})} \left[p\left(\mathrm{VM}_p^{ij}\right) \cdot t\left(\mathrm{VM}_p^{ij}\right) \cdot x_p^{ij}\right], \tag{3}$$

$$m_{\mathrm{transmission}} = \sum_{j,k=1}^{n_{\mathrm{dc}}} \sum_{i,l=1}^{n_{\mathrm{type}}} \sum_{p,q=1}^{n(\mathrm{VM}^{ij})} \left[p'\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot v\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot x_p^{ij} \cdot x_q^{lk}\right], \tag{4}$$

$$m_{\mathrm{total}} = m_{\mathrm{rental}} + m_{\mathrm{transmission}}. \tag{5}$$

m_{rental} of formula (3) represents the rental monetary overhead and $m_{\mathrm{transmission}}$ of formula (4) represents the data transmission monetary overhead. m_{total} of formula (5) represents the total monetary overhead. $p(\mathrm{VM}_p^{ij})$ denotes the rental price of VM_p^{ij}. $t(\mathrm{VM}_p^{ij})$ denotes the rental duration time of VM_p^{ij}. $p'(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk})$ denotes the data transmission price between VM_p^{ij} and VM_q^{lk}.

4. Proposed Approach

In this section, we illustrate our resource allocation approach AIMING. In real-world, most real-time communications happen in a region, such as a country. Thus, the communications in a region are far more frequent than the communications among regions. Based on this assumption, we optimize the total latency according to regions. We first divide the datacenters into K regions; then resource is allocated for each region to minimize the total latency.

4.1. Datacenter Preprocess. In this section, we preprocess N datacenters by dividing the datacenters into K regions. The distribution of datacenters is heterogeneous. Thus, for convenience, we divide the datacenters according to the end-to-end latency $L = t(\mathrm{ping})/v(\mathrm{packet})$. We use k-means clustering to divide the datacenters into K regions. Given a set of datacenter latency observations $(\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_N)$, where each observation is a N-dimensional real vector, k-means clustering aims to partition the N observations into K ($\leqslant N$) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$ so as to minimize the within-cluster sum of squares (WCSS). Hence, the objective is to find

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{L \in \mathbf{S}_i} \left\| L - \mu_i \right\|^2, \tag{6}$$

where μ_i is the mean of points in S_i.

Figure 4 is an example of datacenters clustering when the number of datacenters N equals 5. We divide the datacenters into K regions (K is equal to 2). The latency between two

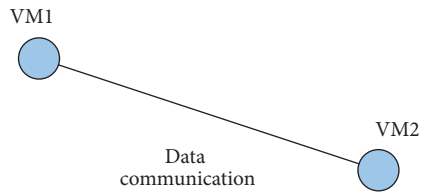

Figure 2: Data communication mode between two VMs.

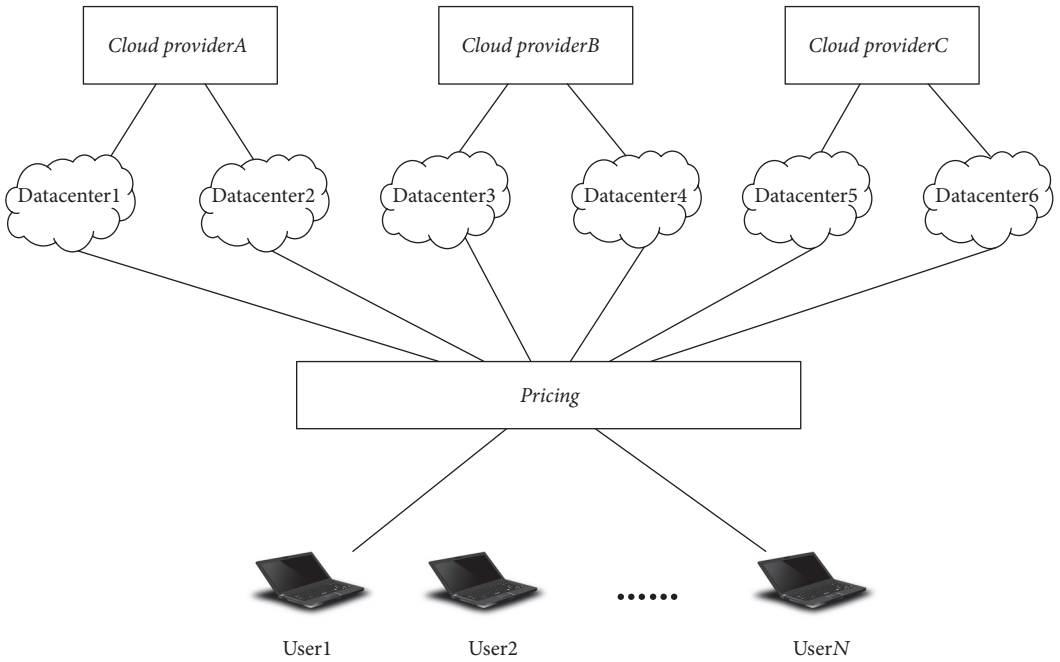

Figure 3: Pricing mechanism of federated-cloud.

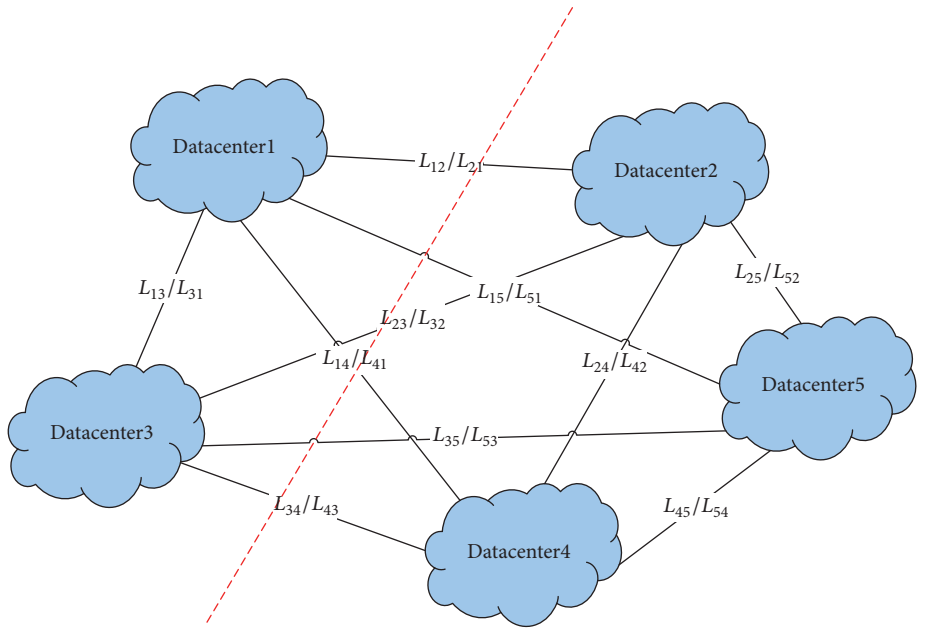

Figure 4: An example of datacenters clustering when N is equal to 5 and K is equal to 2.

Table 1: End-to-end latency of datacenters.

Latency	Datacenter1	Datacenter2	Datacenter3	Datacenter4	Datacenter5
Datacenter1	L_{11}	L_{12}	L_{13}	L_{14}	L_{15}
Datacenter2	L_{12}	L_{22}	L_{23}	L_{24}	L_{25}
Datacenter3	L_{13}	L_{23}	L_{33}	L_{34}	L_{35}
Datacenter4	L_{14}	L_{24}	L_{34}	L_{44}	L_{45}
Datacenter5	L_{15}	L_{25}	L_{35}	L_{45}	L_{55}

datacenters is nondirectional, which means that $L_{ij} = L_{ji}$, $1 \leqslant i$, and $j \leqslant 5$. We get the sets $\mathbf{S} = \{S_1, S_2\}$ which are the two parts shown in Figure 4. The latency set of datacenters is shown in Table 1. Each observation is a 5-dimensional real vector, $L_i = \{L_{i1}, L_{i2}, L_{i3}, L_{i4}, L_{i5}\}$, $1 \leqslant i \leqslant 5$.

4.2. Latency Optimization of Datacenter Region. In this section, we minimize the total latency of each region by placing VMs. The region total latency optimization problem is modeled as binary quadratic programming. For region S_r, N_r represents the total need number of datacenters of region S_r. The number of selected datacenters of region S_r is represented as $N_r' \leqslant N_r$. The purpose is to select N_r' datacenters from N_r and place VMs to minimize the total latency of the region. We assume that the capacity need of each region for customer request is known to us. The capacity performance of a given instance type i is denoted as C_i. Hence, the objective of region S_r can be shown as follows:

$$\min \quad l_{\text{total}}^{r} = \min \sum_{j,k=1}^{N_r} \sum_{i,l=1}^{n_{\text{type}}} \sum_{p,q=1}^{n(\text{VM}^{ij})} v\left(\text{VM}_p^{ij}, \text{VM}_q^{lk}\right) \cdot \frac{t_{jk}\left(\text{ping}\right)}{v\left(\text{packet}\right)} \cdot x_p^{ij} \cdot x_q^{lk}. \tag{7}$$

Users can specify the following types of deployment constraints as follows:

$$\sum_{j=1}^{N_r} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \geq N^{ir}{}_{\text{vm}} \quad i = 1, 2, \ldots, n_{\text{type}}, \tag{8}$$

$$\sum_{i=1}^{n_{\text{type}}} C_i \cdot \left(\sum_{j=1}^{N_r} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \right) \geq C^{r}{}_{\text{need}}, \tag{9}$$

$$N^{\min j}{}_{\text{vm}} \leq \sum_{i=1}^{n_{\text{type}}} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \leq N^{\max j}{}_{\text{vm}} \quad j = 1, 2, \ldots, N_r, \tag{10}$$

$$C^{\min j}{}_{\text{vm}} \leq \sum_{i=1}^{n_{\text{type}}} C_i \cdot \left(\sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \right) \leq C^{\max j}{}_{\text{vm}} \quad j = 1, 2, \ldots, N_r, \tag{11}$$

$$m_{\text{total}}^{r} = m_{\text{rental}}^{r} + m_{\text{transmission}}^{r} \leq \text{Budget}_r. \tag{12}$$

(1) *VM hardware configuration constraints are expressed by restricting the number of each instance type and total infrastructure capacity need*. Formula (8) is the number of VMs of each instance type constraint. $N^{ir}{}_{\text{vm}}$ is the minimum VMs number of instance type i of region S_r. Formula (9) is the total infrastructure capacity constraint of region S_r. $C^{r}{}_{\text{need}}$ is the minimum infrastructure capacity need of region S_r.

(2) *Load balancing constrains are expressed as VM number and infrastructure capacity need of each datacenter*. Formula (10) is the VM number constraint of each datacenter. Users can determine this type of constraint of datacenter j by $N^{\min j}{}_{\text{vm}}$ and $N^{\max j}{}_{\text{vm}}$, the minimum and maximum VMs number of datacenter j, $1 \leqslant j \leqslant N'{}_r$. Formula (11) is the infrastructure capacity need constraint of each datacenter. Users can determine this type of constraint of datacenter j by $C^{\min j}{}_{\text{vm}}$ and $C^{\max j}{}_{\text{vm}}$, the minimum and maximum capacity need of datacenter j, $1 \leqslant j \leqslant N'{}_r$.

(3) *Monetary overhead constraint is expressed as budget of VM rental overhead and data transmission overhead*. Budget_r of formula (12) is the monetary constraint of region S_r.

Moreover, the number of selected datacenters for region S_r is N_r'. And we should note that each VM has to be exactly one instance type and placed in exactly one cloud.

Lemma 1. *The region total latency optimization problem is NP-hard.*

Proof. We prove the NP-hardness by reduction from 0-1 Knapsack problem. Given a knapsack instance with n items, each with a weight w_i and a value v_i, along with a maximum weight capacity W, we create the region total latency optimization problem as follows. The number of datacenters N_r and N_r' are equal to 1. The CPU of each type of VM instance is equal to 0, which means no infrastructure capacity herein. And the VM rental price is particular. There are n links among VMs corresponding to the items. Assuming that the latency of each link between VMs is L_i, then $-L_i$ is corresponding to value v_i and data transmission price is corresponding to weight w_i. The budget constraint is the only limitation of the datacenter, which is corresponding to maximum weight capacity W. Finally, 0-1 knapsack problem can be reduced to the region total latency optimization problem. Thus, the region total latency optimization problem is NP-hard. □

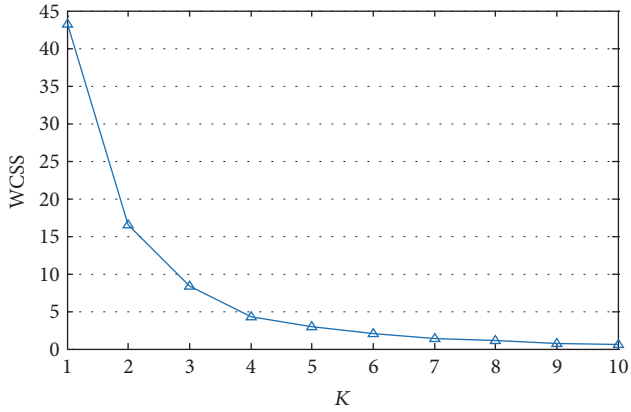

Figure 5: The WCSS of datacenter preprocess.

Table 2: VM performance parameters.

Cloud provider	Instance type	CPU	Memory (GiB)
Azure	D1 v2	1	3.5
	D2 v2	2	7
	D3 v2	4	14
	D4 v2	8	28
Google	n1-standard-1	1	3.75
	n1-standard-2	2	7.5
	n1-standard-4	4	15
	n1-standard-8	8	30
Amazon	t2.small	1	2
	t2.large	2	8
	t2.xlarge	4	16
	t2.2xlarge	8	32

There are multiple model languages and solvers which can be used to solve the optimization problem. We choose the AMPL as modelling language. AMPL has good support for sets and its syntax is close to mathematical notation. Since AMPL can also be used for various types of optimization problems for quadratic programming formulations described above, we use the CPLEX solver.

5. Performance Evaluation

The AIMING is based on federated-cloud environment that is hard to be implemented in practice due to its large network. We evaluate AIMING by simulation and compare it with several approaches. The evaluation shows total latency and rental overhead under various numbers of VMs and datacenters. Parameters analysis includes budget and deviation of VM number. Extensive evaluation results show that AIMING is efficient in total latency reduction.

5.1. Experimental Setup. (1) Using python with PyCharm and based on real-world latency data trace, the data preprocess runs in a variety of realistic settings. We set up a federated-cloud environment with 22 datacenters and 3 cloud providers including Amazon Web Service, Microsoft Azure, and Google Cloud Platform. The 9 datacenters of Microsoft Azure are South USA, Central USA, Sao Paulo, Dublin, Amsterdam, East Asia, Japan West, Singapore, and East Australia. The 4 datacenters of Google are Taiwan, Europe, central USA, and East USA. The other 9 datacenters of Amazon are Oregon, Virginia, California, Sao Paulo, Dublin, Frankfurt, Tokyo, Singapore, and Sydney. We use the interdatacenters latency dataset measured by Mansouri and Buyya [35] from Cloud Laboratory of The University of Melbourne. They have measured the latency between 22 datacenters for 8 hours. The packet size is between 64 bytes and 1 KB and the interval time between each packet is 4 seconds. The latency dataset is measurable and reliable because the cloud providers have their own fiber links interconnecting all major regions. Each network topology of datacenter is fat-tree structure.

Figure 5 shows the WCSS with variety K. The number of datacenters N is equal to 22. In Figure 5, the WCSS is below 5 after K equals 4. Thus, we choose 4 as the value of K and get the clustering result as follows.

Cluster 1 contains 5 datacenters: Azure Dublin, Azure Amsterdam, Google Europe, Amazon Dublin, and Amazon Frankfurt.

Cluster 2 contains 8 datacenters: Azure East Asia, Azure Japan West, Azure Singapore, Azure Australia East, Google Taiwan, Amazon Tokyo, Amazon Singapore, and Amazon Sydney.

Cluster 3 contains 7 datacenters: Azure South USA, Azure Central USA, Google Central USA, Google East USA, Amazon Oregon, Amazon Virginia, and Amazon California.

Cluster 4 contains 2 datacenters: Azure Sao Paulo and Amazon Sao Paulo.

Given that each region has the similar trend, we conduct the latency optimization algorithm on region 1 with 5 datacenters in the following part.

(2) In this section, we use AMPL with CPLEX to implement the latency optimization algorithm. Four VM instance types are considered in the evaluation. For the rental VM configuration, the performance parameters are listed in Table 2. As for the rental price and data transmission price, we use the price from their websites [36–38]. We conduct the latency optimization algorithm on 5 datacenters of cluster 1. For convenience, we define a variable, the number of VMs for each instance type, to replace the total VMs number. Only 4 VM instance types are considered here; then the total number of VMs is 4 multiples.

In order to evaluate the performance of AIMING, we compare it with two approaches in terms of total latency and monetary overhead:

(i) Random: this approach is a random method with the constraints of VM number and capacity need.

(ii) Greedy: this approach selects N_r' datacenters with cheapest VM rental price to allocate VMs [39].

(iii) AIMING: our approach is based on binary quadratic programming.

5.2. Total Latency. In this section, we show the optimal result of total latency with various numbers of VMs and datacenters.

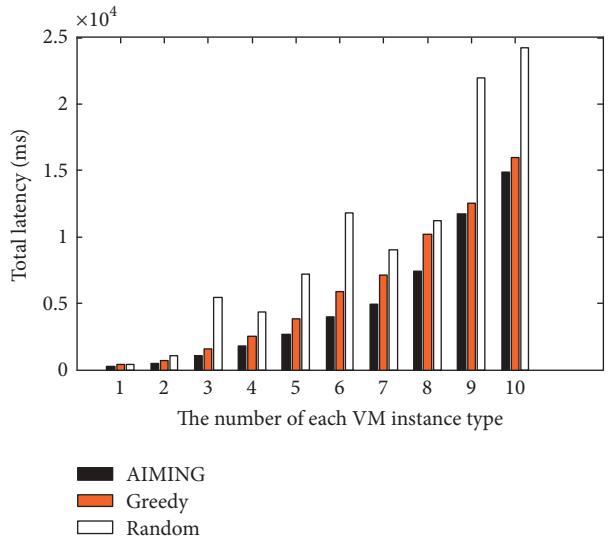

Figure 6: The total latency with various numbers of each VM instance type, which ranges from 1 to 10.

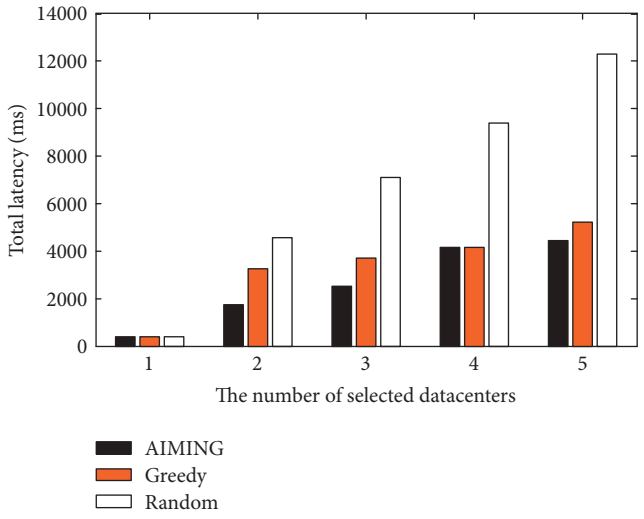

Figure 7: The total latency with various numbers of selected datacenters, which ranges from 1 to 5.

As shown in Figure 6, the total latency of AIMING is better than other two approaches. The total latency of AIMING is about 14906 ms when the number of each VM instance type is 10; however, other approaches (e.g., Random, at about 24198 ms, and Greedy, at about 15977.4 ms) are higher. We set the number of datacenters as 3. The total latency increases along with the number of each VM instance type, which ranges from 1 to 10. However, the difference value between AIMING and Greedy also increases along with the number of VMs. This is because when the number of each VM instance type increases, the data transmission need increases vastly, which leads to higher difference value of total latency.

Figure 7 shows the relationship between total latency and number of selected datacenters. The number of each VM instance type is equal to 5, which means that there are 20 VMs

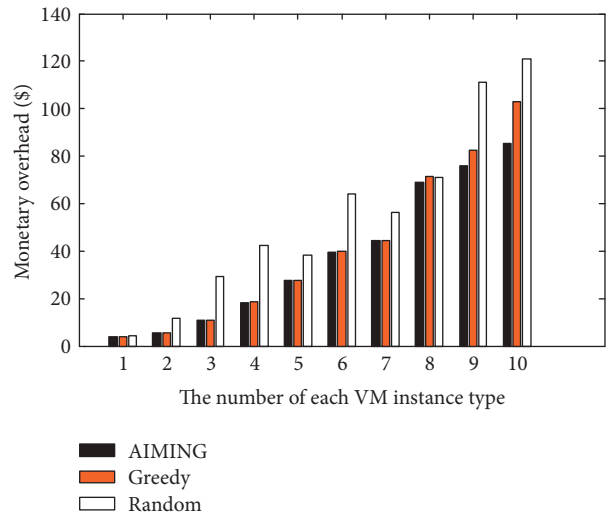

Figure 8: The monetary overhead with various numbers of each VM instance type, which ranges from 1 to 10.

to allocate. When the number of selected datacenters is equal to 1, only the datacenter can be chosen. Then, AIMING, Greedy, and Random have the same total latency which is 400 ms. The Greedy ignores the transmission latency because it often chooses the cheapest datacenters. The AIMING can decrease a maximum 46.1% latency time of Greedy approach when the number of selected datacenters is 2.

5.3. Parameter Analysis

(1) Monetary Overhead and Number of Each VM Instance Type. Figure 8 describes the relationship between monetary overhead and the number of each VM instance type. We set the number of selected datacenters as 3 which is the same as Figure 6. The Greedy and AIMING nearly have the same rental overhead until the number of each VM instance type equals 6. The reason is as follows: when the number of each VM instance type is not large enough, the VM rental overhead plays a more important role in monetary overhead than data transmission overhead. After the number of each VM instance type equals 6, the data transmission overhead occupies a large proportion and AIMING is obviously better than Greedy and Random. When the number of each VM instance type is 10, the monetary overheads of AIMING, Greedy, and Random are $85.275, $103.1247, and $120.8789.

(2) Monetary Overhead and Number of Selected Datacenters. The relationship between monetary overhead and selected datacenters number is shown in Figure 9. The number of VM for each instance type is equal to 5, which is the same with Figure 7. When the number of selected datacenters is 3, the AIMING starts to show its advantage in monetary reduction more than Greedy. When the number of selected datacenters is equal to 3, it will lead to frequent data interaction among VMs and data transmission overhead becomes the major expense. The monetary overhead of the three approaches increases along with the number of selected datacenters.

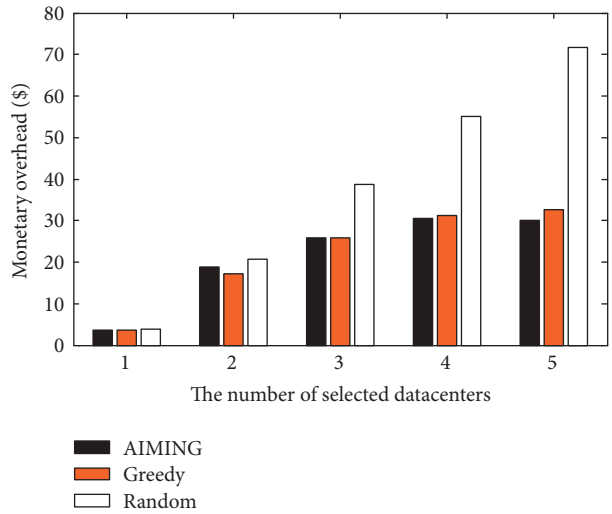

Figure 9: The monetary overhead with various numbers of selected datacenters, which ranges from 1 to 5.

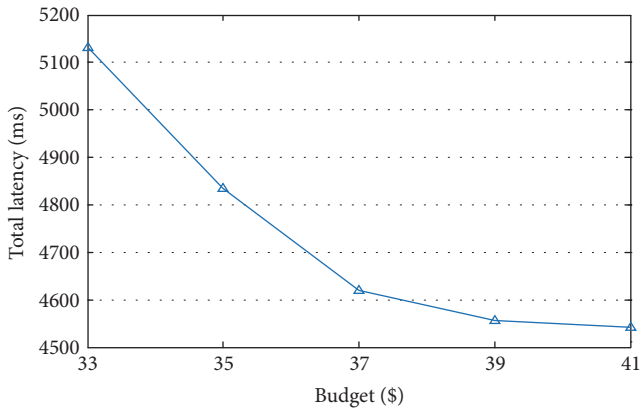

Figure 10: The total latency with budget ranges from $33 to $41.

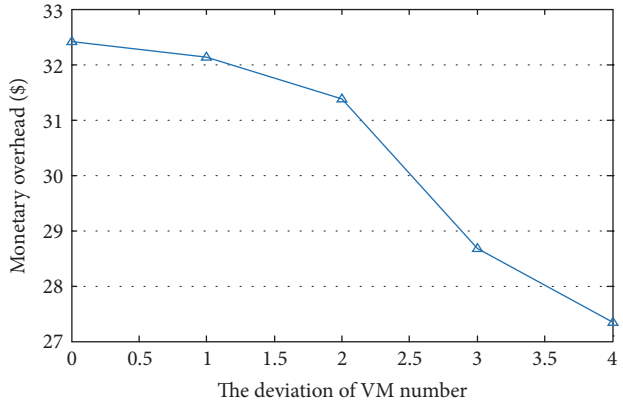

Figure 11: The monetary overhead with various deviations of VM number ranges from 0 to 4.

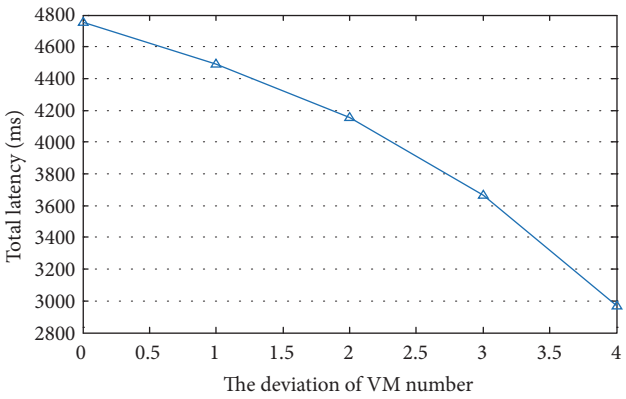

Figure 12: The total latency with various deviations of VM number ranges from 0 to 4.

When the number of selected datacenters is equal to 5, the monetary overheads of AIMING, Greedy, and Random are $30.1725, $32.538, and $71.6723.

(3) Total Latency and Budget. The relationship of total latency and budget is shown in Figure 10. The total latency decreases when budget increases. The number of selected datacenters is set to be 5. The number of each VM instance type is 5, which means that there are 20 VMs to be allocated. We control the budget ranging from $33 to $41, and the difference of latency decreases from 5130.8 ms to 4542.9 ms. The decline is caused by relaxing budget constraint. With the increase of budget, the VM allocation choice scope becomes larger. The better allocation solution will appear and lead to lower total latency.

(4) Monetary Overhead and Deviation of VM Number. Using deviation of VM number, we discuss the VM number constraint of each datacenter. In Figures 11 and 12, the number of each VM instance type is equal to 5 and the number of selected datacenters is 5, which that means there are 20 VMs to be allocated in 5 datacenters. To explain the deviation of VM number, we take an example: if the deviation of VM number is 0, the constraint of VM number of each datacenter is no more than 5 and no less than 5, which means that the constraint of VM number of each datacenter is 5. If the deviation of VM number is equal to 1, the constraint of VM number of each data center is no more than 6 and no less than 4. In Figure 11, the monetary overhead decreases from $32.4114 to $27.3404 when the deviation of VM number increases from 0 to 4. It is because the increasing of deviation will enlarge the solution scope. Hence, the curve shows an impressive decline after the deviation is equal to 2.

(5) Total Latency and Deviation of VM Number. In Figure 12, the total latency decreases along with increase of VM number deviation. The reason is as follows: when the deviation of VM number increases, the VM allocation scope becomes larger. Thus, better allocation solution will decrease the total latency, and the curve declines after deviation equals 2. The total latency decreases from 4755 ms to 2967.8 ms when the deviation of VM number increases from 0 to 4.

6. Conclusion

In this paper, we propose an optimization approach named AIMING to optimize the total latency in federated-cloud scenario. The approach includes two steps: datacenter preprocess and latency optimization. The first step is based on k-means clustering, and quadratic programming is used for the latter step. With real latency among datacenters, VM rental price, and data transmission price as input, we use python and AMPL to conduct the evaluation. The evaluation shows that AIMING can reduce the total latency and monetary overhead efficiently. Besides, we also do a lot of parameter analyses to show more details and impact factors.

Notations

${VM_p}^{ij}$:	pth VM of ith type in jth datacenter
$n(VM^{ij})$:	The VM number of ith instance type in jth datacenter
$p({VM_p}^{ij})$:	The rental price of pth VM of ith instance type in jth datacenter
$p'({VM_p}^{ij}, {VM_q}^{lk})$:	The data transmission price between ${VM_p}^{ij}$ and ${VM_q}^{lk}$
$l({VM_p}^{ij}, {VM_q}^{lk})$:	The latency between ${VM_p}^{ij}$ and ${VM_q}^{lk}$
$t({VM_p}^{ij})$:	The rental duration time of ${VM_p}^{ij}$
$v({VM_p}^{ij}, {VM_q}^{lk})$:	The transmission data volume between ${VM_p}^{ij}$ and ${VM_q}^{lk}$
v(packet):	The packet size of ping
t_{jk}(ping):	The latency of ping from jth datacenter to kth datacenter
n_{dc}:	The number of datacenters which are taken into consideration
n_{type}:	The number of the VM instance type
m_{total}:	Total monetary overhead
m_{rental}:	VM rental monetary overhead
$m_{\text{transmission}}$:	Data transmission monetary overhead
l_{total}:	The total latency of datacenters
${x_p}^{ij}$:	If ${VM_p}^{ij}$ is selected to be rented, ${x_p}^{ij} = 1$; otherwise, ${x_p}^{ij} = 0$
N_r:	Total number of datacenters of region S_r
S_r:	rth datacenter region of clustering
${N'}_r$:	The number of selected datacenters of S_r
${N^{ir}}_{\text{vm}}$:	Minimum number of instance types i of region S_r
${C^r}_{\text{need}}$:	Minimum infrastructure capacity need of region S_r
${N^{\min j}}_{\text{vm}}$:	The minimum VMs number of datacenter j, $1 \leqslant j \leqslant {N'}_r$
${N^{\max j}}_{\text{vm}}$:	The maximum VMs number of datacenter j, $1 \leqslant j \leqslant {N'}_r$
${C^{\min j}}_{\text{vm}}$:	The minimum infrastructure capacity need of datacenter j, $1 \leq j \leq {N'}_r$
${C^{\max j}}_{\text{vm}}$:	The maximum infrastructure capacity need of datacenter j, $1 \leq j \leq {N'}_r$
Budget_r:	The monetary constraint of region S_r.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work presented in this study is supported by NSFC (61602054) and Beijing Natural Science Foundation (4174100).

References

[1] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online Allocation of Virtual Machines in a Distributed Cloud," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 238–249, 2017.

[2] T. Guo, P. Shenoy, K. K. Ramakrishnan, and V. Gopalakrishnan, "Latency-aware virtual desktops optimization in distributed clouds," *Multimedia Systems*, pp. 1–22, 2017.

[3] N. Grozev and R. Buyya, "Regulations and latency-aware load distribution of web applications in Multi-Clouds," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 3261–3280, 2016.

[4] A. Aral and T. Ovatman, "Network-aware embedding of virtual machine clusters onto federated cloud infrastructure," *The Journal of Systems and Software*, vol. 120, pp. 89–104, 2016.

[5] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 963–971, Orlando, Fla, USA, March 2012.

[6] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM '10)*, pp. 1–9, San Diego, Calif, USA, March 2010.

[7] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[8] K. Li, H. Zheng, and J. Wu, "Migration-based virtual machine placement in cloud systems," in *Proceedings of the 2013 IEEE 2nd International Conference on Cloud Networking, CloudNet 2013*, pp. 83–90, USA, November 2013.

[9] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[10] C. Gao, H. Wang, L. Zhai, Y. Gao, and S. Yi, "An energy-aware ant colony algorithm for network-aware virtual machine placement in cloud computing," in *Proceedings of the 22nd IEEE International Conference on Parallel and Distributed Systems, ICPADS 2016*, pp. 669–676, chn, December 2016.

[11] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *IEEE Transactions on Services Computing*, 2016.

[12] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1842–1850, can, May 2014.

[13] C. Tian, A. Munir, A. X. Liu et al., "Multi-tenant multi-objective bandwidth allocation in datacenters using stacked congestion control," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, Ga, USA, May 2017.

[14] R. Yu, G. Xue, X. Zhang, and D. Li, "Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, Ga, USA, May 2017.

[15] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 28–36, Toronto, Canada, May 2014.

[16] Z. Wu and H. V. Madhyastha, "Understanding the latency benefits of multi-cloud webservice deployments," *Computer Communication Review*, vol. 43, no. 1, pp. 13–20, 2013.

[17] I. Narayanan, A. Kansal, A. Sivasubramaniam et al., "Towards a Leaner Geo-distributed Cloud Infrastructure," in *Proceedings of the USENIX Workshop on Hot Topic in Cloud Computing (HotCloud*, pp. 1–7, 2014.

[18] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358–367, 2012.

[19] Y.-H. Lee, K.-C. Huang, M.-R. Shieh, and K.-C. Lai, "Distributed resource allocation in federated clouds," *The Journal of Supercomputing*, vol. 73, no. 7, pp. 3196–3211, 2017.

[20] P. Yi, H. Ding, and B. Ramamurthy, "Budget-Optimized Network-Aware Joint Resource Allocation in Grids/Clouds Over Optical Networks," *Journal of Lightwave Technology*, vol. 34, no. 16, pp. 3890–3900, 2016.

[21] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proceedings of the 32nd IEEE Conference on Computer Communications, IEEE INFOCOM 2013*, pp. 854–862, Italy, April 2013.

[22] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generation Computer Systems*, vol. 50, pp. 62–74, 2015.

[23] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2010*, pp. 513–517, Australia, May 2010.

[24] W. Wang, D. Niu, B. Li, and B. Liang, "Dynamic cloud resource reservation via cloud brokerage," in *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013*, pp. 400–409, USA, July 2013.

[25] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proceedings of the 6th ACM Symposium on Cloud Computing, ACM SoCC 2015*, pp. 111–124, USA, August 2015.

[26] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, "CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, 2015.

[27] D. Ardagna, M. Ciavotta, and M. Passacantando, "Generalized Nash Equilibria for the Service Provisioning Problem in Multi-Cloud Systems," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 381–395, 2017.

[28] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed Online Resource Allocation in Multi-Tier Distributed Cloud Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2556–2570, 2017.

[29] F. Palmieri, U. Fiore, S. Ricciardi, and A. Castiglione, "GRASP-based resource re-optimization for effective big data access in federated clouds," *Future Generation Computer Systems*, vol. 54, pp. 168–179, 2016.

[30] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[31] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, USA, April 2016.

[32] S. Wang, W. Su, X. Zhu, and H. Zhang, "A Hadoop-based approach for efficient web service management," *International Journal of Web and Grid Services*, vol. 9, no. 1, pp. 18–34, 2013.

[33] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.

[34] Y. Guo, S. Wang, K. S. Wong, and M. H. Kim, "Skyline service selection approach based on QoS prediction," *International Journal of Web and Grid Services*, vol. 13, no. 4, pp. 425–447, 2017.

[35] Y. Mansouri and R. Buyya, "To move or not to move: Cost optimization in a dual cloud-based storage architecture," *Journal of Network and Computer Applications*, vol. 75, pp. 223–235, 2016.

[36] Microsoft Azure Price, https://azure.microsoft.com/en-us/pricing/.

[37] Amazon EC2 Price, https://aws.amazon.com/ec2/pricing/.

[38] Google Cloud Platform Price, https://cloud.google.com/pricing/.

[39] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical Programming*, vol. 1, pp. 127–136, 1971.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 9641712, 7 pages
https://doi.org/10.1155/2018/9641712

Research Article

Detection Performance of Packet Arrival under Downclocking for Mobile Edge Computing

Zhimin Wang,[1] Qinglin Zhao ,[1] Fangxin Xu,[1] Hongning Dai ,[1] and Yujun Zhang [2]

[1] *Faculty of Information Technology, Macau University of Science and Technology, Avenida Wei Long, Taipa, Macau*
[2] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*

Correspondence should be addressed to Qinglin Zhao; zqlict@hotmail.com

Received 8 November 2017; Revised 11 January 2018; Accepted 29 January 2018; Published 28 February 2018

Academic Editor: Anna Kobusinska

Mobile edge computing (MEC) enables battery-powered mobile nodes to acquire information technology services at the network edge. These nodes desire to enjoy their service under power saving. The sampling rate invariant detection (SRID) is the first downclocking WiFi technique that can achieve this objective. With SRID, a node detects one packet arrival at a downclocked rate. Upon a successful detection, the node reverts to a full-clocked rate to receive the packet immediately. To ensure that a node acquires its service immediately, the detection performance (namely, the miss-detection probability and the false-alarm probability) of SRID is of importance. This paper is the first one to theoretically study the crucial impact of SRID attributes (e.g., tolerance threshold, correlation threshold, and energy ratio threshold) on the packet detection performance. Extensive Monte Carlo experiments show that our theoretical model is very accurate. This study can help system developers set reasonable system parameters for WiFi downclocking.

1. Introduction

Mobile edge computing (MEC) [1] aims to provide computing resources and information technology services at the network edge. In MEC, various battery-powered mobile nodes (such as smartphone) will access these resources and services via MEC application servers such as LTE base station and wireless access point (AP). These battery-powered nodes desire to enjoy their service under power saving.

In this paper, we assume that a number of battery-powered nodes access an AP (acting as an MEC application server) via a WiFi network. These devices adopt a novel algorithm called sampling rate invariant detection (SRID) [2] for power saving. SRID is the first downclocking mechanism (adopted in WiFi). With SRID, a node detects one packet arrival at a downclocked rate. Upon a successful detection, the node reverts to a full-clocked rate to receive the packet immediately. For each detection, there are two types of typical errors: miss-detection (i.e., the AP sends a packet but the node does not detect it) and false-alarm (i.e., the AP sends nothing but the node detects a packet mistakenly). To ensure that a node acquires its service immediately, the detection performance (namely, the miss-detection probability and the false-alarm probability) of SRID is of importance. This paper is concerned with the detection performance. Our contributions are summarized as follows:

(i) To the best of our knowledge, this paper is the first one to theoretically analyze the detection performance of WiFi downclocking. Our theoretical model characterizes the crucial impact of SRID attributes (e.g., tolerance threshold, correlation threshold, and energy ratio threshold) on the packet detection performance (i.e., the miss-detection probability and the false-alarm probability).

(ii) We run extensive Monte Carlo experiments to verify that our theoretical model is very accurate. We show that as the downclocked rate decreases, the false-alarm probability increases significantly, which will lead to a serious adverse impact on packet detection.

This study can help system developers set reasonable system parameters for WiFi downclocking.

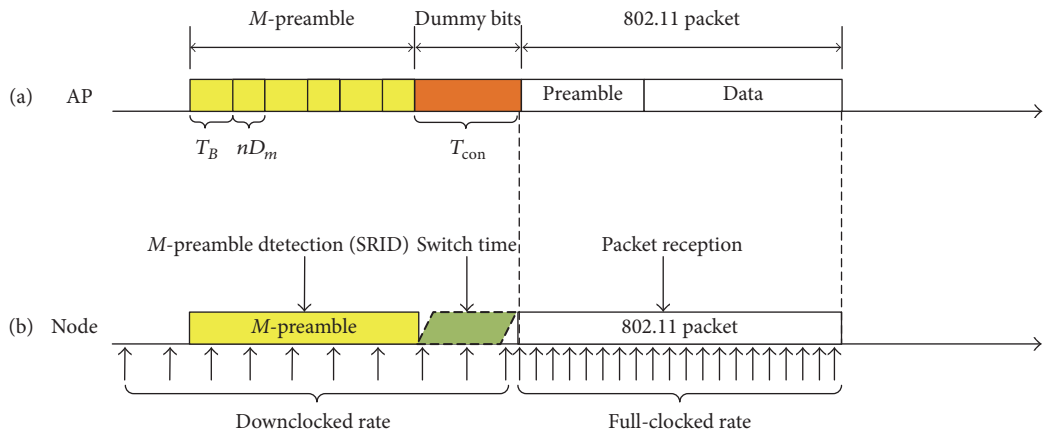

Figure 1: (a) AP's M-preamble transmission and (b) node's M-preamble detection in SRID.

So far, downclocking has received great attention [2–13]. Among the most relevant works, [2] is the first paper that brought downclocking to low-power WiFi networks and proposed the SRID algorithm, which was considered as one of the most classical amendments on power saving of 802.11 protocols [12, 13]. In WiFi, the dominant source of energy consumption is the idle listening operation [7, 8], where a node needs to frequently detect unpredictably arriving packets or assess a clear channel with high power. Therefore, SRID reduced the power consumption by allowing a WiFi node to downclock its sample rate in idle listening mode. SASD [3] was proposed to reduce the power consumption in SRID further by allowing nondestination nodes in idle listening to enter a doze state. AS-MAC [4] was proposed to avoid contention and reduce delay by asynchronously scheduling the wake-up time of neighboring nodes via a downclocking mechanism for wireless sensor networks. SloMo [5] was proposed to allow WiFi nodes to operate their radios at lower clock rates when receiving and transmitting at low bit rates. Sampleless [6] allowed energy-constrained devices to scale down their sampling rates regardless of channel conditions. The above works mainly focused on the hardware implementation of the downclocking mechanism or evaluated its performance via simulation. In contrast, this paper is the first one to model the impact of downclocking on the packet detection performance theoretically.

The rest of this paper is organized as follows. Section 2 gives an overview of SRID. Section 3 theoretically analyzes the detection performance of SRID. Section 4 presents Monte Carlo results that reveal the crucial impact of SRID attributes on the detection performance. Section 5 concludes this paper.

2. Overview of SRID

In the downclocking mechanism, one basic problem is how to detect unpredictably arriving packets at a downclocked rate, so that the node can revert to a full-clocked mode to receive the arriving packets.

The SRID that adopts the downclocking mechanism is designed for WiFi networks. With the help of Figure 1, we specify how SRID works. Assume a WiFi network consisting of one access point (AP) and a number of nodes. The AP is always in the active mode, while each node is in the downclocked mode by default. When the AP has a packet to transmit toward a node, the operations of the AP and the node are as follows.

(i) The AP first transmits an additional preamble called M-preamble, and then a sequence of dummy bits, and finally a conventional 802.11 packet. Here, the M-preamble is used to notify the node of the arrival of an expected packet. The dummy bits are used to provide a guard interval that allows the node to revert to the full-clocked mode from the downclocked mode.

(ii) The node continuously detects its M-preamble via self-correlation and then reverts to the full-clocked mode upon a successful detection.

In the next two subsections, we detail the construction and the detection of an M-preamble.

2.1. Construction of M-Preamble. In SRID, an M-preamble consists of C ($C \geq 2$) duplicated versions of a complex gold sequence (CGS), the length of each CGS sequence being $T_B + nD_m$. Figure 1(a) shows an example of M-preamble, where $C = 3$. Thus the total length of M-preamble T can be expressed as

$$T = C\left(T_B + nD_m\right), \tag{1}$$

where T_B represents the minimum length of the CGS (used for M-preamble). The integer n represents the address of the node in SRID, which is assigned by the AP. $1/D_m$ is the minimum downclocking factor of radio hardware. For example, assume that the full-clocked frequency is 20 MHz. Then the minimum downclocked frequency is $20 * (1/D_m)$ MHz.

2.2. Detection of M-Preamble. In SRID, a node continuously performs self-correlation to detect its M-preamble. Assume that a node operates with a downclocking factor of $1/D \in [1/D_m, 1]$.

Let $z(k)$ denote the sampling value of the node at the sampling point k.

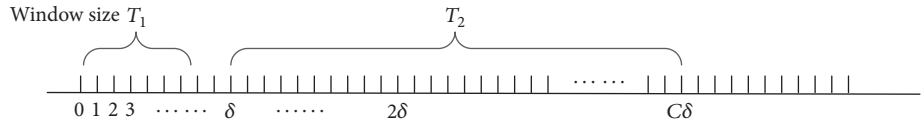

Figure 2: Illustration of C, δ T_1 and T_2.

Let $R(k)$ denote the self-correlation result of the node at the sampling point k. To detect its M-preamble, at each sampling point k, the node with address n performs the self-correlation between the latest T_1 samples and the previous T_1 samples (offset by δ). Therefore, $R(k)$ can be calculated by

$$R(k) = \sum_{i=k}^{k+T_1-1} z(i)\, z(i-\delta), \tag{2}$$

where $T_1 = T_B/D$ is the size of the self-correlation window (in sampling points) and $\delta = (T_B + nD_m)/D$ is the number of sampling points of a CGS when the downclocking factor is $1/D$. Note that T_1 and δ are shown in Figure 2.

Let $E(k)$ denote the energy level at sampling point k, which can be calculated by

$$E(k) = \sum_{i=k}^{k+T_1-1} |z(i)|^2. \tag{3}$$

We say that an M-preamble is successfully detected if the total number of successfully detected points, N_s, is greater than H_1T_2; namely,

$$N_s \geq H_1T_2, \quad H_1 \in (0,1), \tag{4}$$

where H_1 is the tolerance threshold and $T_2 = (C-1)\delta$ is the total number of sampling points (from the 2nd CGS to the C-th CGS), as shown in Figure 2.

We say that a sampling point is detected successfully if the following two conditions are satisfied.

Condition 1. At sampling point k, the correlation result $|R(k)|$ normalized by $E(k)$ is between H and $1/H$; namely,

$$H < \frac{|R(k)|}{E(k)} < H^{-1}, \quad H \in (0,1), \tag{5}$$

where $H \in (0,1)$ is a predefined threshold.

Condition 2. At sampling point k, the energy ratio (in dB) of $E_a(k)$ and $E_a(k - C\delta)$ exceeds a threshold H_s; namely,

$$10 \cdot \log_{10} \frac{E_a(k)}{E_a(k - C\delta)} \geq H_s, \tag{6}$$

where $E_a(k) = T_1^{-1}E(k) + (1 - T_1^{-1})E_a(k-1)$ represents a moving average of energy level, with a window size equal to T_1. The reason of introducing Condition 2 is to reduce the probability that Condition 1 is satisfied but no M-preambles are transmitted.

3. Detection Performance Analysis

In this section, focusing on the downlink traffic from the AP to nodes, we theoretically analyze the crucial impact of SRID attributes (namely, tolerance threshold H_1, correlation threshold H, and energy ratio threshold H_s) on the detection performance.

Due to the downclocked rate and the noise, each SRID detection result is associated with four mutually exclusive minievents: (a) successful detection: AP sends an M-preamble and the node detects it successfully, (b) miss-detection: AP sends an M-preamble but the node does not detect it, (c) false-alarm: AP does not send an M-preamble but the node detects it mistakenly, and (d) Null: AP does not send an M-preamble and the node detects nothing. To study the detection performance, we only need to calculate the successful detection probability P_d = Prob(successful detection) and the false-alarm probability P_{fa} = Prob(false-alarm), because Prob(miss-detection) = 1 − Prob(successful detection) and Prob(Null) = 1 − Prob(false-alarm).

We note that each detection result is determined depending on whether the AP sends an M-preamble. Below, we introduce two competing hypotheses:

$$\begin{aligned} &\mathcal{H}_0 : z(k) = n(k), \quad k = 0, 1, \ldots, C\delta \\ &\mathcal{H}_1 : z(k) = hx(k) + n(k), \quad k = 0, 1, \ldots, C\delta, \end{aligned} \tag{7}$$

where $\mathcal{H}_0$ is referred to as the null hypothesis (i.e., AP does not send an M-preamble to a node) and $\mathcal{H}_1$ as the alternative hypothesis (i.e., AP sends an M-preamble to a node). Under hypothesis $\mathcal{H}_0$, at the sampling point k, the node receives the noise, and therefore its sample value is $z(k) = n(k)$, where $n(k)$ is the Gaussian white noise. Under hypothesis $\mathcal{H}_1$, at the sampling point k, the node receives the M-preamble signal and the noise, and therefore its sampling value is $z(k) = hx(k) + n(k)$, where $x(k)$ represents the sampling value on the M-preamble and h represents the channel coefficient.

3.1. Expression of P_d. We now express P_d. According to (4), we have

$$P_d = P\{N_s \geq H_1T_2 \mid \mathcal{H}_1\}. \tag{8}$$

The sampling process is a Bernoulli process, where a sampling point is marked success if Conditions 1 and 2 (specified in Section 2.2) are satisfied. Therefore, the number of successfully detected points in T_2 trials, N_s, follows a binomial distribution. Thus P_d is expressed by

$$P_d = \sum_{i=H_1T_2}^{T_2} C_{T_2}^{i} \left(P_1P_{\text{ER1}}\right)^i \left(1 - P_1P_{\text{ER1}}\right)^{T_2-i}, \tag{9}$$

//Input: SNR, $C, T_B, nD_m, H_1, H, H_s, h, D, T, T_2$
//Output: P_d
//We run the code below for 100000 times.
(1) $i \longleftarrow 0$
(2) while ($i < 100000$)
(3) Generate a CGS randomly
(4) $x(0, 1, \ldots, T) \longleftarrow [\text{CGS}, \text{CGS}, \ldots, \text{CGS}]_{1\times C}$
(5) $y(0, 1, \ldots, T) \leftarrow \text{awgn}(x(1, \ldots, T), \text{SNR}, \text{“measured”})$
(6) $z(0, 1, \ldots, C\delta) \leftarrow$ Sample y every D points
(7) Calculate $R(0, 1, \ldots, C\delta)$ and $E(0, 1, \ldots, C\delta)$
(8) $N_1 \leftarrow$ the total number of sampling points that satisfy Condition 1.
(9) $P_1(i) \longleftarrow \dfrac{N_1}{C\delta}$
(10) $N_2 \leftarrow$ the total number of sampling points that satisfy Condition 2.
(11) $P_{\text{ER1}}(i) \longleftarrow \dfrac{N_2}{C\delta}$
(12) $i \longleftarrow i + 1$
(13) end
//We first calculate $\text{avg}(P_1)$ and $\text{avg}(P_{\text{ER1}})$, then P_d.
(14) $P_d \longleftarrow \sum_{i=H_1T_2}^{T_2} C_{T_2}^{i} [\text{avg}(P_1)\text{avg}(P_{\text{ER1}})]^i [1 - \text{avg}(P_1)\text{avg}(P_{\text{ER1}})]^{T_2-i}$

Algorithm 1: Calculation of P_d based on the Monte Carlo method.

where P_1 is the probability of Condition 1 being satisfied under $\mathscr{H}_1$ and P_{ER1} is the probability of Condition 2 being satisfied under $\mathscr{H}_1$.

Expression of P_1. According to (5), P_1 can be written as

$$P_1 = P(\mathscr{H}_1; \mathscr{H}_1) = P\left\{H < \frac{|R(k)|}{E(k)} < H^{-1} \mid \mathscr{H}_1\right\}. \quad (10)$$

In (10), $P(\mathscr{H}_i; \mathscr{H}_j)$ represents the probability of deciding $\mathscr{H}_i$ when $\mathscr{H}_j$ is true. Let U_1 denote the normalized correlation result at sampling point k under $\mathscr{H}_1$. Then U_1 can be expressed as $U_1 = \sum_{i=k}^{k+T_1-1} |(hx(i) + n(i))(hx(i-\delta) + n(i-\delta))| / \sum_{i=k}^{k+T_1-1} |hx(i) + n(i)|^2$. Thus P_1 is expressed by

$$P_1 = P\left\{H < U_1 < H^{-1}\right\}. \quad (11)$$

Note that U_1 is complicated, because it is a function of $2T_1$ random variables (i.e., $n(k-\delta), n(k+1-\delta), \ldots, n(k+T_1-1-\delta)$ and $n(k), n(k+1), \ldots, n(k+T_1-1)$). In Section 3.3, we calculate P_1 via the Monte Carlo method [14].

Expression of P_{ER1}. According to (6), P_{ER1} can be written as

$$P_{\text{ER1}} = P\left\{10 \cdot \log_{10} \frac{E_a(k)}{E_a(k - C\delta)} \geq H_s \mid \mathscr{H}_1\right\}. \quad (12)$$

Under $\mathscr{H}_1$, $E_a(k)$ and $E_a(k - C\delta)$ are expressed as follows.

$$\begin{aligned} E_a(k) &= T_1^{-1} E(k) + \left(1 - T_1^{-1}\right) E_a(k-1) \\ &= T_1^{-1} \sum_{i=k}^{k+T_1-1} |hx(i) + n(i)|^2 \\ &\quad + \left(1 - T_1^{-1}\right) E_a(k-1) \\ E_a(k - C\delta) &= T_1^{-1} E(k - C\delta) \\ &\quad + \left(1 - T_1^{-1}\right) E_a(k - C\delta - 1) \\ &= T_1^{-1} \sum_{i=k}^{k+T_1-1} |n(i)|^2 \\ &\quad + \left(1 - T_1^{-1}\right) E_a(k-1). \end{aligned} \quad (13)$$

Note that $E(k - C\delta) = \sum_{i=k}^{k+T_1-1} |n(i)|^2$, because AP transmits one M-preamble (which consists of $C\delta$ sampling points only) for each packet, and thereby the node only receives the noise before the M-preamble. Similar to P_1, we can calculate P_{ER1} via the Monte Carlo method.

3.2. Expression of P_{fa}. We now express P_{fa}. Similar to P_d, P_{fa} can be expressed as follows:

$$\begin{aligned} P_{\text{fa}} &= P\{N_s \geq H_1T_2 \mid \mathscr{H}_0\} \\ &= \sum_{i=H_1T_2}^{T_2} C_{T_2}^{i} (P_2P_{\text{ER2}})^i (1 - P_2P_{\text{ER2}})^{T_2-i}, \end{aligned} \quad (14)$$

where $P_2 = P(\mathscr{H}_1; \mathscr{H}_0) = P\{H < |R(k)|/E(k) < H^{-1} \mid \mathscr{H}_0\}$ is the probability of Condition 1 being satisfied under $\mathscr{H}_0$ and $P_{\text{ER2}} = P\{10 \cdot \log_{10}(E_a(k)/E_a(k - C\delta)) \geq H_s \mid \mathscr{H}_0\}$ is the probability of Condition 2 being satisfied under $\mathscr{H}_0$.

3.3. Calculation of P_d *and* P_{fa} *via Monte Carlo Method.* In the previous two subsections, we give expressions of P_d and P_{fa}. However, they involve $2T_1$ random variables and therefore are hard to solve. Below we adopt the Monte Carlo method [14] to calculate them. Algorithm 1 lists the computation process in

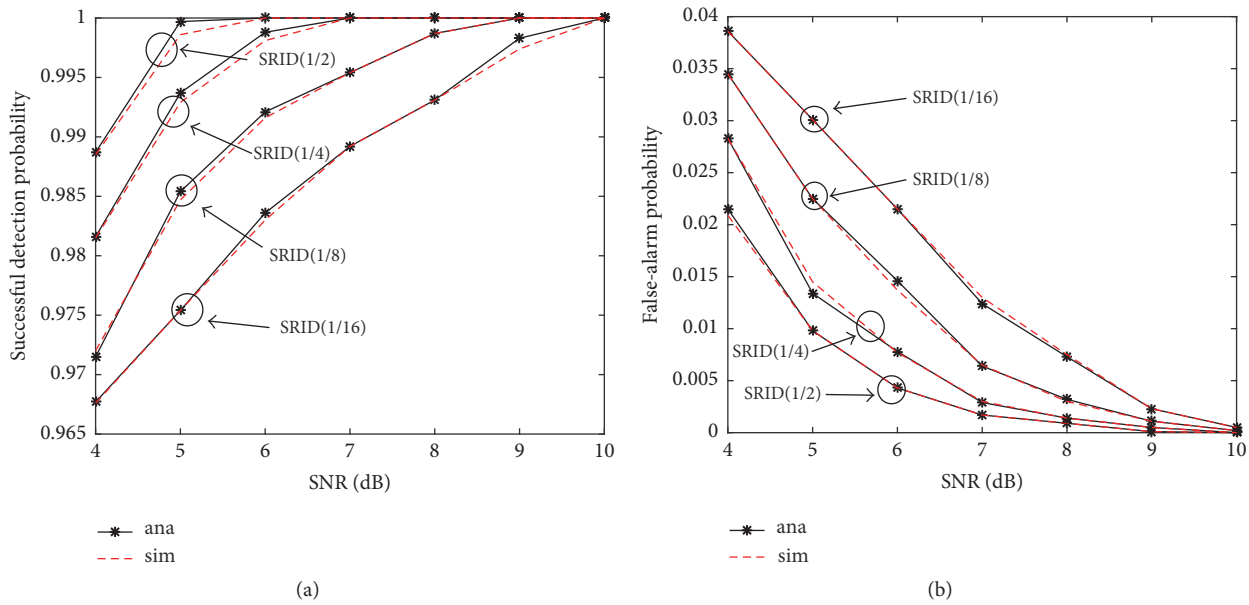

Figure 3: (a) P_d and (b) P_{fa} vary when the SNR varies.

terms of P_d, which is given by (9). Similarly, we can calculate P_{fa}.

In Algorithm 1, we input the SRID parameters and output the value of P_d. In the algorithm, we run Monte Carlo experiment for 100000 times. We now detail each experiment.

(i) In lines (3) to (4), we generate an M-preamble of T samples points, which simulates the AP's M-preamble transmission.

(ii) In line (5), we invoke the Matlab function, awgn(·), to simulate the additive white Gaussian noise (AWGN) channel and then the node's received signal is the result that the AP's M-preamble signal passes through the AWGN channel.

(iii) In line (6), we obtain the downclocked sampling sequence $z(\cdot)$ under the downclocking factor of $1/D$.

(iv) In line (7), we calculate self-correlation result $R(\cdot)$ and the energy level $E(\cdot)$.

(v) In lines (8) to (9), we calculate P_1 in this experiment.

(vi) In lines (10) to (11), we calculate P_{ER1} in this experiment.

Finally, after we finish 100000 runs, we first calculate the average of P_1, avg(P_1), and the average of P_{ER1}, avg(P_{ER1}), and then calculate P_d, as shown in line (14).

4. Model Verification

In this section, we present the Monte Carlo results to illustrate the crucial impact of SRID attributes and SNR on the detection performance (namely, the successful detection probability P_d and the false-alarm probability P_{fa}). The default parameter settings are set by [2] and are shown in Table 1.

Table 1: Parameter settings in simulation.

Parameters	Description	Values
C	Number of CGS	3
T_B	Basic length	64 sampling points
nD_m	Additional length	64 sampling points
H_1	Tolerance threshold	0.6
H	Correlation threshold	0.9
H_s	Energy ratio threshold	4 dB
SNR	Signal-to-noise ratio	9 dB
h	Channel coefficient	1

In Figures 3 and 4, each Monte Carlo result is on average over 100000 runs. In addition, we use "SRID($1/D$)" to denote the SRID detection with the downclocking factor of $1/D$. In all figures, the labels "ana" and "sim", respectively, denote the theoretical and simulation results.

Figures 3(a) and 3(b), respectively, plot P_d and P_{fa} as the SNR varies, when $1/D = 1/2, 1/4, 1/8, 1/16$. From Figure 3, we have the following observations.

(i) Given $1/D$, P_d increases and P_{fa} decreases gradually as the SNR increases.

(ii) Given SNR, as $1/D$ decreases, P_d decreases slightly while P_{fa} increases significantly. For example, for SNR = 5 dB, when $1/D$ decreases from 1/2 to 1/16, P_{fa} grows from 0.0098 to 0.0301 significantly, while P_d just drops from 0.9997 to 0.9754 slightly. This will lead to a serious adverse impact on packet detection.

(iii) The detection performance is almost perfect (i.e., $P_d = 1$ and $P_{\text{fa}} = 0$) when SNR = 10 dB, which is easy to achieve in real environments [15].

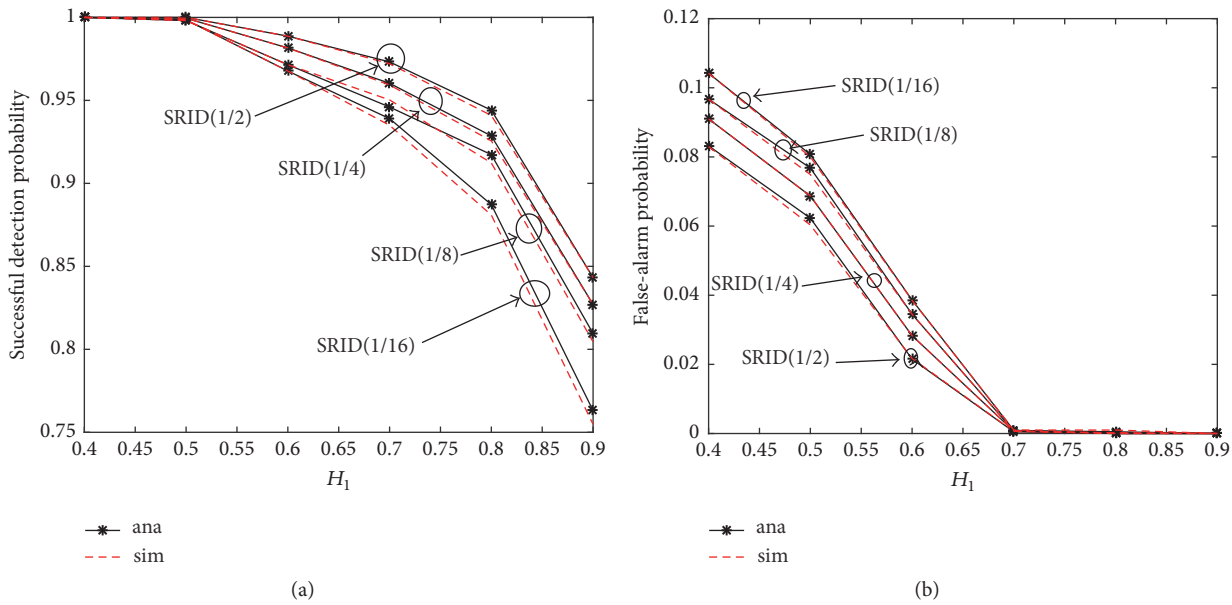

FIGURE 4: (a) P_d and (b) P_{fa} vary when H_1 varies.

Figures 4(a) and 4(b), respectively, plot P_d and P_{fa} as H_1 varies, when $1/D = 1/2, 1/4, 1/8, 1/16$. From Figure 4, we have the following observations.

(i) Given $1/D$, as H_1 increases, P_d always decreases, but P_{fa} first decreases to 0 and then remains unchanged. The reason is as follows: increasing H_1 will decrease the successful detection probability from (9) as well as the false-alarm probability from (14).

(ii) Give H_1, as $1/D$ decreases, P_d decreases significantly, while P_{fa} decreases gradually.

Finally, from these figures, the close match between the theoretical and simulation curves manifests that our performance model is very accurate.

5. Conclusion

In mobile edge computing, various battery-powered mobile nodes desire to acquire information technology services at the network edge under power saving. WiFi downclocking is such a promising technique. In this paper, we investigate a novel WiFi downclocking technique called SRID and first theoretically study the impact of SRID attributes (namely, tolerance threshold, correlation threshold, and energy ratio threshold) on the detection performance of packet arrival. This study is helpful in designing better WiFi downclocking protocols.

Disclosure

Qinglin Zhao is the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the Macao FDCT-MOST Grant 001/2015/AMJ, Macao FDCT Grants 056/2017/A2 and 005/2016/A1, National Science Foundation of China (61672500), and Program of International S&T Cooperation (2016YFE0121500).

References

[1] https://en.wikipedia.org/wiki/Mobile_edge_computing.

[2] X. Zhang and K. G. Shin, "E-mili: energy-minimizing idle listening in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 9, pp. 1441–1454, 2012.

[3] T. Xiong, J. Yao, J. Zhang, and W. Lou, "It Can Drain Out Your Energy: An Energy-Saving Mechanism Against Packet Overhearing in High Traffic Wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 16, no. 7, pp. 1911–1925, 2017.

[4] B. Jang, J. B. Lim, and M. L. Sichitiu, "An asynchronous scheduled MAC protocol for wireless sensor networks," *Computer Networks*, vol. 57, no. 1, pp. 85–98, 2013.

[5] Lu F., Voelker G. M., Snoeren A. C. SloMo: downclockingWiFi communication[C]// Usenix Conference on Networked Systems Design and Implementation. 2013:255-268.

[6] W. Wang, Y. Chen, L. Wang, and Q. Zhang, "Sampleless Wi-Fi: Bringing low power to Wi-Fi communications," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1663–1672, 2017.

[7] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless wakeups revisited: energy management for VoIP over Wi-Fi smartphones," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys '07)*, pp. 179–191, June 2007.

[8] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues," *Wireless Networks*, vol. 14, no. 6, pp. 745–768, 2008.

[9] P. Serrano, A. De La Oliva, P. Patras, V. Mancuso, and A. Banchs, "Greening wireless communications: Status and future directions," *Computer Communications*, vol. 35, no. 14, pp. 1651–1661, 2012.

[10] X. Zhang and K. G. Shin, "Gap Sense: lightweight coordination of heterogeneous wireless devices," in *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM '13)*, pp. 3093–3101, April 2013.

[11] W. R. Dieter, S. Datta, and W. K. Kai, "Power reduction by varying sampling rate," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, pp. 227–232, usa, August 2005.

[12] Y. Cui, X. Ma, H. Wang, I. Stojmenovic, and J. Liu, "A survey of energy efficient wireless transmission and modeling in mobile cloud computing," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 148–155, 2013.

[13] H. A. Omar, K. Abboud, N. Cheng, K. R. Malekshan, A. T. Gamage, and W. Zhuang, "A Survey on High Efficiency Wireless Local Area Networks: Next Generation WiFi," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2315–2344, 2016.

[14] https://en.wikipedia.org/wiki/Monte_Carlo_method.

[15] J. T. Bevan et al., "An Integrated 802.11a Baseband and MAC Processor," in *IEEE ISSCC Digest*, 2002.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7243875, 23 pages
https://doi.org/10.1155/2018/7243875

Research Article

Centralized Connectivity for Multiwireless Edge Computing and Cellular Platform: A Smart Vehicle Parking System

Aamir Shahzad,[1] **Jae-young Choi**,[2] **Naixue Xiong**,[3] **Young-Gab Kim**,[1] **and Malrey Lee**[4]

[1]*Department of Computer and Information Security, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea*
[2]*Department of Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[3]*Department of Mathematics and Computer Science, Northeastern State University, 611 N. Grand Ave, Tahlequah, OK 74464, USA*
[4]*Center for Advanced Image and Information Technology, School of Electronics & Information Engineering, Chonbuk National University, 664-14 1Ga Deokjin-dong, Jeonju, Chonbuk 561-756, Republic of Korea*

Correspondence should be addressed to Young-Gab Kim; alwaysgabi@sejong.ac.kr and Malrey Lee; mrlee@chonbuk.ac.kr

Received 13 October 2017; Revised 2 December 2017; Accepted 17 December 2017; Published 20 February 2018

Academic Editor: Shangguang Wang

This study takes an intuitive step to develop the user-convenient smart vehicle parking system (SVPS), a smart system able to manage the massive crowd of vehicles during parking searching and do the better jobs of parking reservation and management, with the shorter-path processing tactics. For that, this study inclusively employed the mapping strategy, where the system parking points are prevalent, to assist the users to get the parking information fast and conveniently. This study is comprised of the several parking points systematically spread over the several locations and traceable over the available graphical map, and the overall information is easily accessible using smart devices. For parking information, a smart web application which is another important module of this study is designed, with which the SVPS system's registered users are able to access all the services provided for smart vehicle parking searching and reservation in efficient and reliable ways. An integrated network approach, RFID (radio frequency identification) and wireless sensors network (WSN), called RF-WSN, is employed to retrieve the real-time information from the installed and configured sensor devices in RFID-WSN network.

1. Introduction

With the enhancements of the Internet of Things (IoT) that have been made in several areas of human lives, the traffic control and management systems are predominated, and for that, several intelligent solutions are implemented to manage the massive traffic crowd in metropolitan areas. Typically, a long time ago, the human movements are massively increasing towards the big cities, in almost all over the world; therefore the traffic issues are also increasing at the same time more especially in the cases of finding the parking places. Notably, the case of finding a parking place in a cowed city that has available parking lots is a big challenge, which has been faced by vehicles owners. The recently approaches employed to find the vehicle parking lots are manual; therefore the case of finding parking lots in the metropolitan city in a massive traffic crowd is a big task. Some parking places used computerized systems to keep the checks on vehicles that are going in/out from the parking places and monitoring systems are mainly facilitated through the surveillance systems (e.g., CCTV). However, it depends on the person; if he/she is lucky, he/she could find the parking space upon arrival at the parking place; thus this is very transparent that most of the time is always wasted in finding parking place, if available. Also, in most cases, the vehicle's owner always finds the parking place far from his/her destination; consequently this is not an optimal approach, even in case of predefined parking, which wastes time and vehicle fuel and other energy consumptions [1–3]. For enhancement through the employing of modern technologies, such as the

Internet, wireless, and cellular communications, numerous interactive and noninteractive systems have been introduced for vehicle parking reservations online before the time to search for the destination. This is somehow good but there is no surety to get the confirmation, through the cellular devices, of parking place at the same when required [1, 4, 5].

The smart parking system is also considered as a part of the smart city project, where the advanced technology such as the Internet of Things has been ubiquitously deployed in various sectors, to make the user lives convenient by providing their required services. Thereby, through its use, users or vehicle drivers can able to find a parking lot in any part of the city and will reserve the vacant parking lot according to their predilections, through Internet connectivity, using of electronic reachable devices or/and cellular devices. The parking fees possibly are made using debit/credit cards at the same parking place using parking payment machines. Accessing through the Internet, the smart online parking applications, or parking systems, the user should precisely get the all available information of parking place, parking spots availabilities, parking reservation upon decision, and fees detail and reliable payment methods; of course, it saves enough time and avoids the massive traffic congestion issues from user perspectives. Moreover, during parking spot selection and at the time of car parking, car management system is also a main part of smart parking system which is deployed efficiently to manage the tasks such as online accessible parking point localization, parking reservation through Internet access, vehicle security through tracing the networked tags, parking barrier control, RFID based parking payments, and others [6, 7]. Usually, two types of management systems, (1) main parking gate tracking system and (2) parking lot tracking system, are addressed under smart parking system. The cars (or other vehicles), during entry and exit, passing by the main parking gate, at each check (check-in or checkout) are observed and managed which resulted in various important services to the drivers, including the availability of vacant parking lots in a parking place and availability to make a reservation online, whereas the presence and absence of each car parking over the indicated parking lot are monitored through reading from installed sensors at that lot. Parking lots are generally equipped with sensors or camera equipment for lot monitoring and processed and managed by lot management module as a part of the smart parking system, which therefore also provides services to their users (vehicle owner) including the lot availability and where to park map guideline. Moreover, the parking lot management module can be varying to parking lot design and area, such as indoor lot design (or monoparking) and outdoor lot design; then selective parking lot's module is addressed to manage based on reading observed via lot's sensor installed [6].

Radio frequency identification (RFID) is commonly employed innovative technology, as an important part of the wireless communication system deployed in various automotive industries, aviation and medical sectors and transportation, and parking management systems and others. The commonly used equipment, which made the RFID transmission through radio waves, is the RFID tags (such as active and passive), tags reader, communication media, computer system encompasses of databases, user interface, middleware, gateway, and so forth [7, 8]. Therefore, the RFID technology has been considered tremendous due to facts of cost reduction and gain system's computation efficiency, deployed in a wide range of technological sectors and applications, employing a specific set of hardware and the proprietary protocols, in order to make communication to reading data accurately from tags possible for RFID reader. In RFID system, the two types of tags are commonly used. (1) Active tag, containing an embedded IC chip, coiled antenna, and built-in power source, carries the low-level energy signals from RFID reader and then transmits the identification or resulted signals back to the reader. (2) Passive tag, another type of RFID system in which tag does not have an internal power source and its use of the energy source of RFID reader usually designed to transmit the high power signal towards an RFID tag, uses same power even to transmit back the modulated signals required. The RFID reader is the main module, used in RFID technology, which spreads the signal around in order to get the information from the tag; however, reader is able to get modulated signals from multiple tags at the same time and then process to the application program running on the system for further processing, including storage. In case of an active tag, the identification signal is transmitted around and continued powering with/without concerning reader's field signals, whereas passive tags are activated while high-level signal is received from RFID reader [7–9]. In practical, RFID technology has been playing rigorous roles and widely developed for the transportation systems, in which the vehicle identification in parking systems is more dominated [6, 10].

In automated parking systems, the RFID technology has important roles in vehicle identifications and automated parking fee payments; RFID system enables the autofast and efficient and secure monitoring during the vehicle check-in and checkout from parking gates (or barriers) through tracking tags from RFID reader, including the gates' controlling and their timing that is specified, in the parking place. Therefore, the vehicles can move in/out from the parking gate in sequential order, with the minor delay, which avoids the congestion issues and the multiple vehicle checks-in and checkouts, and the parking payments are collected without any delay that may require each vehicle to stop at the parking gate, through using of RFID based ticket machines [7, 11]. The RFID is a contactless technology employed for object tracking identification, communicating through radio signals, and thus most appropriate for vehicle parking system due to its several benefits: (1) contactless physical communication, (2) enabling tags placement in a loop, possibility of using even in worst scenarios, and environmental conditions, (3) tags which are available according to the communication system demands in the wide range and sizes, (4) minimal service costs, and (5) minimal error rate in nonline of sight communication and others. For an optimal parking system, RFID base system considered as a solution to well manages the parking lots management issues encountered in existing studies [12]. The major issue, such as application processing issue, when the parking lots are not available, resulted in the parking space management issue during the

application process. This issue could be resolved by keeping out those vehicles which are supposed to check in; however, enough time will probably be spent on searching the parking vacant lots. Another major issue generally happened during short/long disconnection with the main system via Internet access, and these issues can be resolved by keeping the communication continued, perform each transaction, and store in the local system setup updated all the saved data that are recorded, to the main system's storage, while the main system status changed to online. Thus, this is a good approach employed to run also the database on a local system and further shift all information to the main system upon connection established with [11].

In studies [10, 12–16], comprehensive reviews were conducted on the smart parking systems which deployed the today's advanced IoT technology; further some important issues were highlighted that were commonly available in parking systems and the usual challenges that have been faced by the drivers (or users) during searching the optimal ways for the parking in the traffic density in the crowded areas. More advanced, in study [10], smart parking system was built with the mathematical model formed to validate the overall system's performance and parking reservations were performed with the minimal cost, but, at the same time, this study has some disadvantages, including the time wastage consumed during car forwarding scenarios to another parking place. Therefore, here, smart parking system needs special care and acquires efficient solution to resolve the car forward issues to the other parking areas, before the case where the researched parking status is full.

In this proposed study, an inclusive and convenient approach is precisely planned to deploy the automated smart vehicle parking system (SVPS), and the SVPS system is intelligently designed and modeled as a complete solution to regulate and monitor its parking points and other pieces of networked equipment and is considering under public sector regulations. In short, this means that the whole proposed system is directly associated with the public sector or governmental sector property. Therefore, the system development emerges and plays important roles in economic benefits and benefits for the users to get the convenient and cheap parking while comparing the rates with the private sector parking systems. Following are the main objectives this study aims to fulfill (or anticipate):

(1) SVPS system is only accessible for its users; therefore a registration module is designed and available online which facilities the registration process for the new users. Upon registration completion, the users could get the RFID tag installation, as well as the RFID access card and the online login ID and the password.

(2) Upon logging in to the online SVPS system, the parking searching and reservation module is designed, as a part of SVPS system's web application, to made parking search easy using designed mapping strategy and to made parking reservation according to user selection or user request for parking. Through mapping, users will find the near parking points and the short route to reach the destinations. Further, the system is efficient in computing the short travelling route, followed by the input location of the user.

(3) At time to access the parking point and selective parking lot, the embedded RFID tag and designed RFID access card are used in order to access main parking gate. For user (or vehicle) authentication and verification, at parking point entry gate the RFID tag is read and the RFID card is verified wirelessly through installed sensors. Then, one more time, vehicle embedded RFID tag is read at its allocated parking lot, for parking lot verification purposes.

(4) For parking utilization and the corresponded payable amounts, the SVPS system is smart in order to show the overall payment of parking utilized, at per hour basis, online into the user account and the parking payment would possibly be payable using various payment services (i.e., using credit/debit card and monthly based transferring facility). Therefore, the user has no worries about the payment to be made at the time of parking (or parking exit).

In this study, the parking point (PP) is used to represent the parking place where a number of parking lots (PL) are available for vehicle parking and the registered vehicles are represented as users of the system. Moreover, in RFID-WSN system, the installed RFID UHF readers at parking lots are designated as slave-sensor nodes connected with the master-sensor nodes.

The rest of paper is organized as follows: Section 2 conducted a comprehensive survey on the various solutions, systems, and technologies employed in existing vehicle parking systems. The system design and framework are described in Section 3 which with the study fulfills its target goals, and in Section 4, smart parking design is detailed and then set up with the numerous of sensor nodes. Section 5 implements a system where registered users or users of the system are allowed to get access the smart parking services available as part of the system and do the parking reservations. Experimental scenarios are defined and employed to conduct the results of the discussion in Section 6. At the end, Section 7 concluded the overall study works and the directions for future work.

2. Literature Survey

Smart parking system, employing Internet of Things (IoT) technology, was implemented to facilitate the parking reservations [2], accessed through the online parking service, using the cellular devices and other accessible devices like tablets, and further the end-users were be able to see the availability of parking lots before the reservations. As a solution, a valid unique identification number (UID) was assigned to each vehicle and used during the whole online parking reservation process and that number should be further applied to authenticate the vehicles at the main entrance of parking place, using RFID devices installed [2, 7, 17]. In spite of, by considering, notably, the limitations of the existing vehicle parking systems, such as undefined short distance parking, managing of traffic load balancing, instantaneous

query acceptance and rejection, and commercial welfare, the IoT technology has been playing important roles to resolve these issues. In a study [1], a cloud-based smart car parking system is modeled, considering the efficient computations of IoT technology, in which each parking place is deemed as the IoT communication node. The important required information is that the vehicle positions' computation using global positioning system (GPS), the estimation of the distance between the current position of the vehicle and the closed parking places, and parking spaces' availability are examined in real-time manners. The observed information is then simultaneously transmitted back to the cloud data center, where the carried information is stored, managed, and updated and is further accessible by the authorized user's request. For system prototyping, physical IoT Arduino platform is used, and the parking places are fully equipped with the RFID technology [1]. A short message service (SMS) is used, as an intelligent parking solution, considering hardware design, using the TC35i module in a global system for mobile communication (GSM), which avoids staying at other wireless media because of the development cost. Thus, using this service, a message sent by the driver to the system to check the status of the parking lots in the parking area and upon confirmation of available parking, the driver allowed using the parking lot as long as the time mentioned, inside the SMS received from the system [18].

In studies [18–21], the advanced technological frameworks for smart car parking systems are introduced, employing the hybrid network, in wired and wireless sensor networks, which deployed modern RFID technology and IEEE standards for communication. Further, the developed smart parking systems are accessible online using Internet's facility available. Thus, upon the parking reservation and confirmation, the system collects the information of available parking places and the information of vacant lots and transmitted it back to the users who could see parking information, using the smart parking application installed in their smart devices [7, 11]. While entering the assigned parking place, the system's allocated user parking lot would appear on the small displaying parking map, employing a short-range protocol called dedicated short-range communications (DSRC). DSRC protocols have been employing several intelligent transportations systems [22]. Further, in the boundary-parking place, inertial navigation system (INS) has been commonly deployed to direct the vehicle to its originated parking position. Thereby, the system design becomes more efficient and gains high accuracy by the means to make parking system always updated, with the information of parking lots, for example, whether vacant or nonvacant, in a specific parking place, in a real-time manner; thus the time usually required by the users to perform parking reservation would significantly minimize [13]. In another study [19], a new architecture, intelligent parking assistant (IPA), is implemented to manage the public parking points located in various parts of the city. The designed architecture is intelligent in computing information of available parking lots in the street parking areas; thus the drivers, those submitted the request for parking, could see the available parking lots information before few minutes of their arrival to the selective parking points. The IPA design was mainly targeted to manage the parking spot that occupied small coverage area and to keep the vehicle check-in and checkout information always updated in the control unit, through employing of the RFID technology. The RFID devices and the magnetic loops are installed in the parking spots and the IPA gets the parking overall information and keeps the control unit updated, periodically.

In [23], the common car parking issues are considered and then the smart car parking system is proposed, employing of ZigBee network, based on IEEE 802.15.4 specification, suited for personal area networks. In the designed system, a coordinator device establishes the network root and the application layer transmits information quickly back to the controller via Internet, where the information is kept updated in the database. As a part of system's design, web service is created, a useful service, to keep the parking prevalent information always updated, for the newer users (or drivers), to make the reservations for the parking lots if available. Lambrinos and Dosis [15] introduced an architecture design for smart car parking system, employing advanced platform called the Internet of Things (IoT), the middleware layer, and ZigBee wireless sensor network (WSN), and the required reporting services are manipulated through IoT front-end layer 12 which acts as an interactive user interface. However, the study has limitations, due to the employment of reliable application protocol, that is, constrained application protocol (CoAP), in transferring of information from the network setup to the control unit. Bonde et al. [24] proposed a miniature model for automated car parking system, whose designed aim is to control and manage the availability of parking slots for the cars in the parking place, according to the given allocated times. For this, a software application is designed, based on Android platform, which manages the given allocated time to park the cars and the existing time from occupied parking lots. The sensor devices installed at parking entry points regulate the check-in and checkout to/from the individual parking lot. To check the parking lots information, convenient for the users, an LCD integrated with the microcontroller is placed at the parking entry point, which displays the parking lots status, time to time. In [25], a mechanical model is designed for smart parking system comprised of various parking levels and the lifting facility therefore is used to make parking automatic. A recognition system is deployed, which recognizes the cars at the parking place through their license number plates. For vehicles lots detection, inside parking place, ultrasonic sensors are installed with liquid crystal display to visualize the vacant parking lots status [26, 27].

In studies [7, 28–39], numerous automotive vehicle parking systems are designed and modeled according to the time-ongoing requirements of vehicle parking, employing of RFID and wireless sensor networks. For that, sensor devices or sensors mostly comprised into intrusive and the nonintrusive have been employing depending on the constraints, as ubiquitous deployments, such as finance, scalable design area, system reliability, and efficiency, to keep a smart check and balance for the system's usage and processing. Despite intrusive sensors technology, nonintrusive sensors utilization

in various wireless sensor networks is accounted as a cost-effective and easy-to-install solutions, specifically for the video image processing solutions which are more robust compared with the others and are used also in the parking systems to detect the vehicles within the boundary of the parking place. The vehicle detection is carried out using the visual images, as a part of the video image processing system and thus, successful observations (e.g., changes) are made through regulation and comparing the successive frames captured in specific time interval [26, 40]. More advanced various existing car parking issues are considered and after, image processing based solutions are proposed to facilitate the patrons well [26, 41]. For development [26], "RabbitCore® Microcontroller image processing" unit is used, integrating with the closed-circuit television (CCTV), in the designed parking area. The purposes of the integration are to made detection of the parking lots in the parking area and to transmit the collected information (of vacant parking lots) back to the central server, keeping the information updated in the database, through ZigBee wireless sensor network (WSN). The visual facility, as a part of car parking system, is installed at the main entrance of parking and visualizes the complete parking map, the availability of lots of each level, and the convenient shorter-path direction by the implementation of A-Star (A^*) algorithm. Automated ticketing and payment machines, located in different parts of parking boundary, are used during entry at the parking and upon exit from the parking place [42]. In [43], the authors propose an image processing solution, deployed to collect the empty car parking spaces information in car parking area. This proposed solution is used to capture the brown rounded image of each parking space and then further performed the detection for free (or available) parking spaces visualize in the seven-segment display. In another study [44], a mechanism is proposed based on the camera vision technique; the free parking lots are detected by setting the input values, for example, input coordinates of parking point, to the object classifier. For that, cars' images are captured in the parking point, through considering various angles, and then classified as positive and negative images. The positive images are used to examine the car presences and on the other side, the negative images are analyzed to show vacant parking or parking lot available for car parking.

3. Proposed System: Design and Framework

In this study, the main goal is to provide a fully automated smart vehicle parking system (SVPS) which should significantly reduce the workforce, usually required in conventional-existing parking systems, and aims to employ an efficient way to deploy the radio frequency identification (RFID) technology in wireless sensors network (WSN). The WSN configuration provides the potential benefits of the proposed parking system or SVPS system which showed a new development to fulfill the advanced parking system requirements, which are probably required by today's modern smart parking systems [5, 9–12]. Therefore, this study contributed well in achieving its goals; the development is mainly a consideration as a part of the Internet of Things (IoT) platform, where all the system devices are networked as fully automotive devices have self-controlling abilities to interact with the overall system-modules. For that, the SVPS system is comprised of the modules designed and deployed as services offered by the SVPS system: (1) parking appointment module, (2) registration module, (3) installation module, (4) web-searching module includes parking reservation management and parking lot management, and (5) parking design and setup, to fulfill the objectives of proposed study.

In Figure 1 the SVPS system's framework is comprised of four main layers: (1) hardware-sensor layer, (2) network access layer, (3) middleware layer, and (4) user application layer, employed to provide a 2-way communication and the interaction between the SVPS system's modules. More precisely, in the below subsections, the SVPS system define-modules are thoroughly explained followed by these four layers. At the instance, SVPS system's design is entirely computation based on the efficient simulation design, as well the conducted measurements, but is very flexible in the planning of further system extension and system modularity. Further, the SVPS system is a system, absolutely developed, under the regulations of the public sector (or local city governmental level), because of the following:

(1) In future, SVPS system's design and framework will probably take over and regulated the public sector and the further intentions in mind to extend the project as a country-level project.

(2) SVPS system's design and framework are directly/indirectly considered as part of IoT system, that is, IoT ubiquitous smart city project.

3.1. Hardware-Sensor Layer. In SVPS system, at the hardware-sensor layer, incorporating RFID technology and with the integration WSN technology, primarily system's services are stated to perform. Further, importantly, the overall useful information is continuously retrieved from the RFID sensors installed and configured in RFID-WSN system. In Figure 1, SVPS system is comprised of a number of parking points and designed to carry the real-time information (or reading), from the installed RFID sensors, through communicating with gateway system. In each parking point, as a part of SVPS system, the parking lots are installed with RFID readers (or sensors) having specifications as follows: (1) ultrahigh-frequency (UHF) type, the usage available frequency range which is about 860–960 MHz, (2) RFID tag detection, the distance range of about 2 meters, and (3) the ISO 18000-6C standard [45–47]. The air-interface protocol (generation 3) is employed, a way of communication between labeled passive RFID tags and installed RFID UHF readers. More precisely, the each exploited RFID reader has been assumed to have sensing capabilities while usage of WSN technology, but for instance, the SVPS system employed WSN technology, integrating with RFID technology, only for the purposes of converging or wireless coverage extension. Thereby, incorporating WSN technology, the information from RFID sensors can be forwarded to system gateway installed and set up in each parking point and to the central control system. RFID

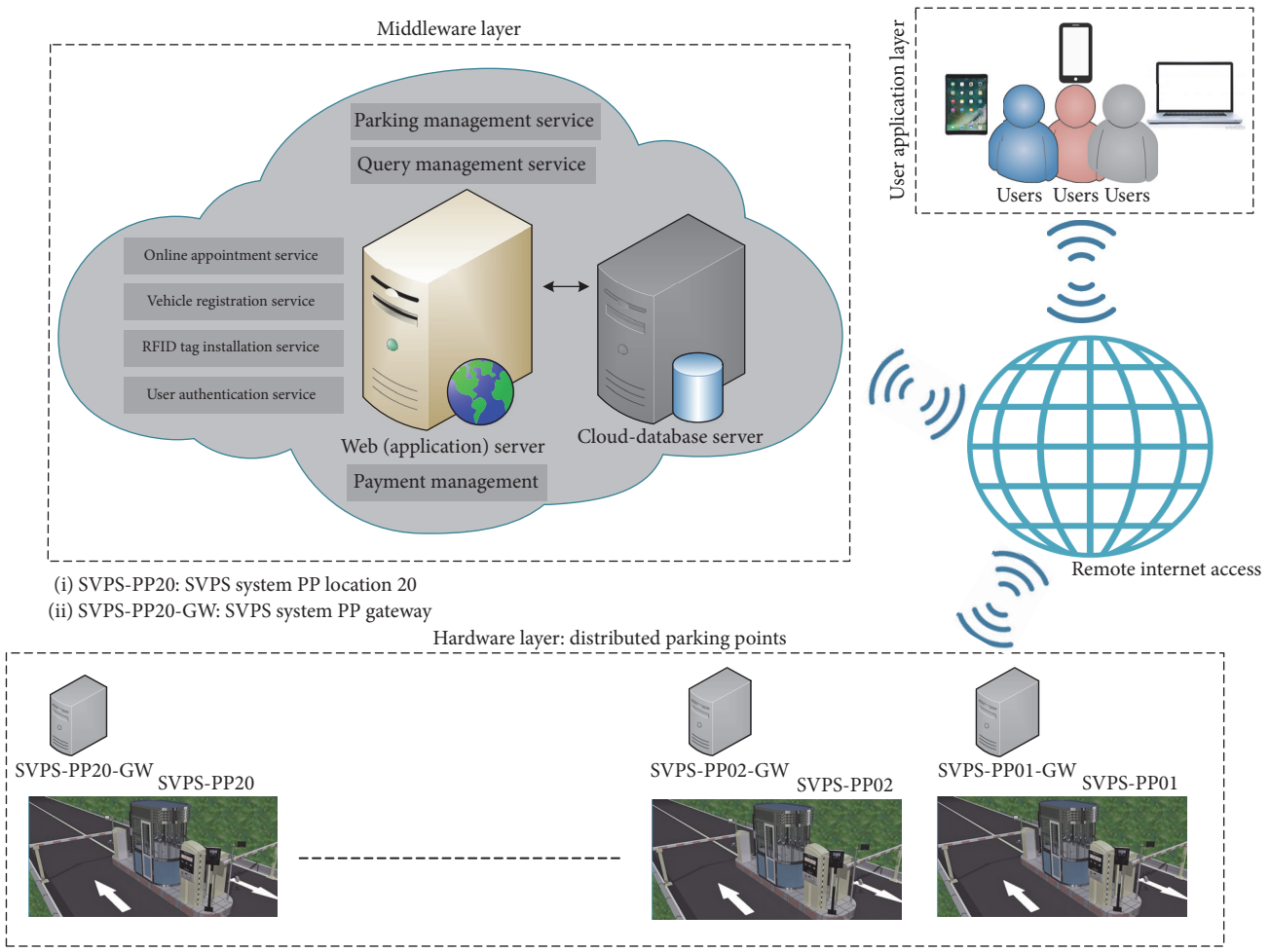

Figure 1: System design and framework.

sensors are installed and positioned the exact center of each parking lot, illustrated in Figure 2. The embedded RFID sensors, also called slave-sensor nodes (SS nodes), are connected directly with the master-sensor node (MS node). MS node is an acquisition node configured to retrieve real-time information from the slave-sensor nodes; however, the number of sensors is limited which is assigned to each master-sensor node. Moreover, the SS nodes are connected directly to the MS node through the wired network, and for this, the interintegrated circuit (I^2C) protocol is employed which is intended for the communication between slave-sensor nodes and a master-sensor node. For local communication, within the premises of parking point (e.g., personal area network), ZigBee wireless network based on IEEE 802.15.4 specifications is used to access and carry information from networked master-sensor nodes (MS nodes) within the distance range of about 10–100 meters. Further, in RFID-WSN network, the repeaters or the anchor nodes (AC nodes) are networked and distributed over the optimal positions in order to retrieve information and to provide coverage to the master-sensor nodes.

3.2. Network Layer. In network setup of each parking point, the slave-sensor nodes are installed and are directly connected, that is, wired network connection, with the master-sensor nodes designated which are wirelessly connected to the local system controller (LSC). Basically, LSC performs the functionalities of gateway system and in each parking point, LSC is designated to carry the traffic from slave-master-sensors nodes and to pass the traffic to the central controller system (CC system) using transmission control protocol (TCP)/Internet protocol (IP), a way of transmission over the Internet. Therefore, the users that made register under SVPS system could get the system's access, using smart electronic devices including GSM/GPRS based cellular devices, directly from the CC system. This means, because of the security issues, the external smart devices exploiting the parking reservations and other system available services are only allowed to access through CC system, not from the selective parking gateway system, only from the central system. However, in few situations or critical situations, the users could access information directly from the target parking point gateway system; this is one of the future considerations

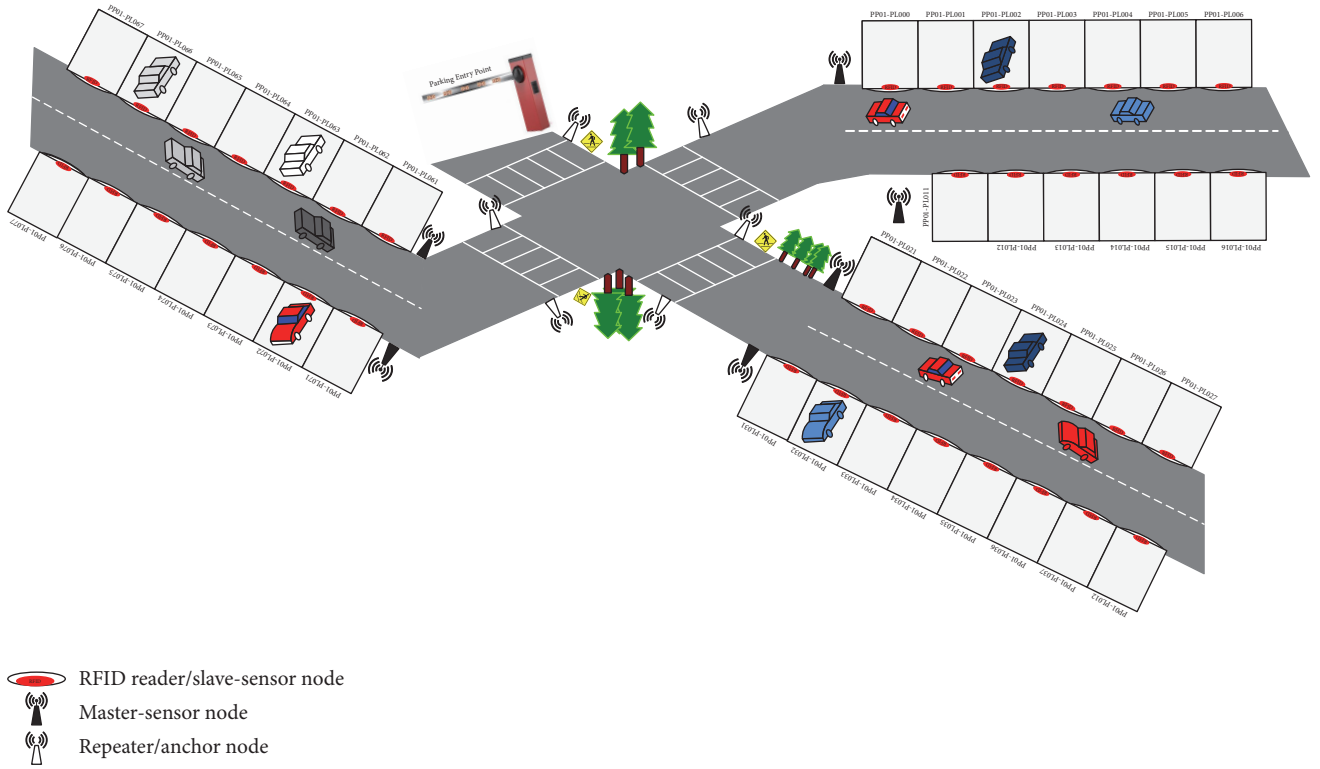

Figure 2: RFID-WSN network setup and configuration.

which will be deployed in presences of security framework that is out of the scope of this study.

3.3. Middleware Layer

3.3.1. Development and Database. In this study, the central controller system is considered as a main centralized controller to access, monitor, and control the overall information from/to the remotely networked parking points which are distributive over the various main locations of the metropolitan city. Therefore, SVPS system performs all required operations, such as the management, monitoring, and controlling operations, through a CC system. SVPS system is entirely designed and developed using the development tools: (1) Microsoft Visual Studio C# as a programming tool, (2) ASP.net for web applications, (3) IIS server for web access, and (4) MySQL database tool. Further, more advanced add-ons are available, for Android and iOS operation systems running on cellular phones, which intended to allow access to the SVPS system online, merely with compatible predominated search browsers such as Microsoft Explorer, Google Chrome, Firefox, and Safari. Thereby, cellular users could able to access the SVPS system online, through compatible browsers, for example, parking slots availability check and parking reservation purposes.

For efficient online searching for parking relevant information and keeping the record always up to date, the cloud-computing design is also a part SVPS system. Thus, SVPS system incorporated a cloud design or also called SVPS-cloud, using Microsoft cloud platform actively employed to fulfill the overall backup storage requirements of the system. However, now, the cloud-computing facility employed in SVPS system is only limited to keep the information backup, the information that is carried from the parking points. In SVPS-cloud, the database design is generic in order to store and keep records of the information individually from each module and is efficient in its computing; therefore, the information can be retrieved inclusively as a result that is computed from distinct modules of SVPS system. For example, those users are only permitted to perform parking reservations previously if they are registered under the system, that is, authorized system's users, and further, the requests made for the parking lots are possibly granted if they are vacant. In order to manage the massive connectivity and network traffic, the SVPS system is developed in such a way that a fully accessible system to cellular devices is running Android or iOS systems, via the Internet. Thereby, the registered users are authorized to access and use the SVPS system's services online using of Internet connectivity available inside the cellular devices.

For user convenience, the parking searching is an important feature employed which provides a reliable and fast way to search the parking points from the location or point where the vehicle is positioned. For that, the Google Maps, using Google application program interface (API), is setup and integrated into SVPS system's design in order to acquire on search a user demandable parking point(s). Thus, as a part of this study, the system uses the Google Maps service to pin the exact parking positions and adds a new feature to locate the

closer parking points, through computing the distance with all the system's available parking points, followed by the user's selection process (i.e., request for parking). As a result, the system will display the most closer (e.g., 3-4 or more) parking points; therefore, the user has options to select the specific parking point among others by his/her interest.

3.3.2. Timing Allocation and Management. The smart system or SVPS system is designed to operate parking allocations, and the parking timing that is required by each vehicle is entirely managed in hourly basis. There is no limitation for the vehicle to fix the time of parking during reservation, that is, the time to leave the allocated parking lot; the SVPS system automatically computes, manages, and notifies the timing, while each authorized vehicle makes entry onto the parking lot and the time of exit, through sensors reading. However, there might be some delay occurring because of the sensors reading, for example, while information is transmitted from sensors to the local system controller (LSC) to central controller system (CC system). For that, SVPS system efficiently measures the timings (e.g., transmission delays) which considerably do not affect the overall system performance. Moreover, entirely, the system is intelligent in order to acquire the timing information, while the vehicle makes entry at the main entrance of selective parking point through sensors data acquisition. For best timing management, enough additional time will be allowed corresponding to the total distance which might be travelled by the vehicle from parking main entrance to the parking lot entrance (allocated). For example, the distance between the parking main entrance and the allocated parking lot computed by the system is approximately 300 meters, which is covered by a vehicle in almost two (or less) minutes, without traffic congestion inside parking point. As a result, the system allocates additional 2 minutes to that vehicle to make entry onto the parking lot. Particularly, the SVPS system is very flexible and efficient in its design, mainly under the requirements of the users request for parking, and it provides the following 3 main features: useful for parking time allocation and management purposes.

(1) Any Time Parking. The registered users have a feature to do parking reservations, accessing the SVPS system online, at any time. For example, on the spot, the user wishes to do parking of his/her vehicle while being closer to any of parking points. By checking the availability of parking lots and then user parking reservation, thus for that selective parking point, the user will allow making entry inside and could stay longer at the parking lot without any time restrictions assign via the system. However, the system manages the timing calculations and the usage of the parking lot, on an hourly basis.

(2) Fixed Time Parking. This smart parking feature allows users to make reservations for parking points, regulated under the system, in case of parking lots availability, prior to the several hours but no more than 48 hours restricted, for a fixed timing. For example, a registered user wishes to make parking reservation in advance, according to the parking point selection, within 48 hours (e.g., Monday at 14:00 pm–22:00 pm). Thus, the system allocates the time session for parking, corresponding to the user selective fixed timing, in advance.

(3) Timed Parking. The users who aim for the parking on the monthly basis use this feature. Thus, the users have the option to do parking reservations, selecting the parking lots, on the monthly basis, and have the option to reserve parking on a daily and weekly basis. For user convenience, the parking reservation is easily to be made just using the online system and by selecting the starting time/day and ending time/day, using the service provided by the year/monthly calendar additionally having a time-date facility.

For the above cases, the CC system is designated to make the reservations, based on the parking lots availability acquired from slave-master sensors, on the users request for parking (RFP). The system is much efficient in its computing, that is, time computing, whenever the vehicle makes entry at the parking main barrier of selective parking point and then at the parking lot allocated from the system. Therefore, the time vehicle arrives at the parking lot; then local system controller (LSC) transmits the information or check-in and checkout status to the CC systems, through the sensors installed at the parking lot.

3.3.3. Registration and Primary Features. The proposed system or SVPS system is designed under the considerations of the fully automated system; for achieving that, SVPS system utilizes the advanced emerging technologies of the current era called the Internet of Things (IoT). Thereby, the system is considered as a fully automated system and reachable to numerous smart devices, such as cellular devices, laptops, and others, using the Internet access. However, the system is not designed with the considerations of security, that is, Internet security as well as the security protection against the potential system's vulnerabilities [48, 49]. So, the comprehensive security design will be a major contribution to this study aimed to draft in nearly future; nevertheless, here in the current stage, this study considers the inclusive smart setup procedure which is useful during the verification process to verify whether or not the vehicle that beings utilizing the parking service is authorized or not. Therefore, in short, a standard new registration method is employed, which will be significant at each stage of parking process (e.g., parking searching and reservation), as an authorized way regulated under SVPS system.

(1) Registration. One of the important steps, among others, is the verification of vehicle being registered under the SVPS system and the authorized owner identification (e.g., valid identification number). At the time of registration, these are basic requirements for each user to be registered; however, a request for SVPS registration could be possible to be made online using of smart appointment module. Therefore, through entering the user complete details, including his/her vehicle details, into the SVPS system, a unique identification number (UID) is generated and then assigned to the vehicle. The UID shows identity of the user authorization and further uses to access the system services overall.

(2) Installation. Upon completion of registration process, for example, all necessary tasks, the smart RFID passive tag called Tag-SVPS (or T-SVPS) and an RFID base chip card called C-SVPS are generated, having an identical unique identification number (e.g., IS0004641V780012104P561S), and printed under the registered vehicle. For each vehicle, at the time after registration, both the T-SVPS and C-SVPS are RFID based and have a similar identification number which will be used to track and acquire the vehicle valid information during system utilizations. The usage of C-SVPS is twofold: (1) at the parking entry it provides an authentication through sensing the identity using RFID sensor installed at the parking barrier; (2) alternatively, it will be very useful in case the embedded T-SVPS is not readable by RFID sensor, due to the network issues such as hardware issues and interference; thus the system permits that vehicle to make entry inside via C-SVPS. However, this situation is not very common and for operation, it requires permission from the CC system.

To make continually interaction with the system, the T-SVPS and C-SVPS having similar identification are always useful, while vehicle identification requires field sensors installed in parking points. For instance, 20 parking points are under the design-consideration of SVPS system and are connected centrally with the CC system, but not with each other. Because of the security issues, the SVPS system only exploits a centralized system (or CC system), intelligently set up to control and manage the information from each parking point.

3.3.4. Smart Access. At the time of registration, a short unique user ID is created, along with the password, for each registered user, that is, it authorized the user of the SVPS system, to access the system services online using of various smart devices. Thus, the system is supportive of the smart devices having various network connections, such as Wifi, GSM, 2G, and 3G, to interact with the system's premises. While logging in to the system online, entering the user ID and secure password on the spot system should get the original accurate position of the user (or user's vehicle location information) through getting the measurements of the positional coordinates, and then the observe position will be dragged onto the map (i.e., Google Maps) showing the designed parking points. Thus, after verification of the user's position and indication, correspondingly the system measures and displays all the real positions of parking points, with having the information of vacant and nonvacant parking lots, over the map. Here, the user can be to select the near parking point which has vacant parking lots, so the request for parking (RFP) is submitted as a further step to make a reservation and then in case of confirmation that will be visualized into user login account.

3.3.5. Smart Payment. Like other smart features available in the SVPS system, the system also provides an efficient and user-convenient method to manage the parking payments; by employing this smart payment method, the user is not restricted to make payment, at the instance, after parking usage or checking out the parking point. As mentioned above, the system has been fully functional on per hour basis; for example, in case the vehicle only spent 30 minutes or more, but less than 1 hour, then the system still computes the parking payment on an hourly basis. However, there will be a margin of almost 10 minutes, after spending an hour, using the vehicle to check out the occupied parking lot. Therefore, after exiting from parking, the total payment will be calculated by the system based on the time the vehicle occupied the parking lot, and the total amount will display inside the user's account. Thus, the user could view the total parking payment or the parking payment history, according to the usage time of the login.

Each time the user uses the system' parking service, the equivalent parking fee (or amount) is calculated and aggregated in the total which should be paid by the user. The user has options to make the payment by the end of the following month, through using various payment methods including the direct account deposit and cheque methods. Further, the users who are permitted to make payments via credit/debit card have to register prior to using the SVPS system. Thereby, the desired total parking payment will be automatically paid at the end of the month or according to the due date. However, these payment methods are now not very new, as an inherent property usually employed by the advanced banking systems, over the world.

3.4. User Application Layer. At the application layer or the abstraction layer, typically all the main employing protocols, interfaces, and system' services are resided and defined to be used by the end-users perspective. Therefore, the SVPS system users could get access to the system available services (or resources), using various smart devices, facilitating through the user application layer.

4. Smart Parking Design and Setup

For instance, the SVPS system's design is comprised of 20 parking points $(\mathrm{PP})_i$, that is, $(\mathrm{PP})_i = \{(\mathrm{PP})_1, (\mathrm{PP})_2, (\mathrm{PP})_3, \ldots, (\mathrm{PP})_n\}$, where n represents the nth parking point, locating at various crowded main parts (or places) of metropolitan city (i.e., Seoul city of South Korea) where most of the popular companies, shopping malls, and other daily working places are usually situated. Each parking point, PP, in total of $(\mathrm{PP})_{(i \leq n)} = 20$, has occupied distinct number of parking lots $(\mathrm{PL})_e$, that is, $(\mathrm{PL})_e = \{(\mathrm{PL})_1, (\mathrm{PL})_2, (\mathrm{PL})_3, \ldots, (\mathrm{PL})_k\}$ and $1 \leq k \leq 150$. Thus, the number of parking lots $(\mathrm{PL})_e$ in each parking point $\mathrm{PP} \in (\mathrm{PP})_i$ is depending on the total area size; for example, the standard maximum area for each designed parking point is not fixed but might surrounded in area of several acres. Similarly, the size (width$*$length) of each parking slot PL is also not defined in a fix size and varies to the vehicle size, but the standard size used is approximately 8 feet by 16 feet, that is, 8 ft wide and 16 ft long. However, this study is not under considerations of that to encompass multiple levels parking; the parking points $(\mathrm{PP})_i$ are designed in open area but covered with the boundary walls. However, each parking point (PP) has different area occupied for vehicle parking with distinct number of parking lots $(\mathrm{PL})_e$, such that $e \leq k = 150$.

Table 1: Designed parking points and mapping description.

Number	PP entitled	Number of gates	Direction	Number of PL	PL size	Number of nodes (slave; master; anchor)	Local mapping
(1)	PP01	2	Northeast	120	8 ∗ 16	(124; 13; var)	Guidance map available
(2)	PP02		Northeast	120		(124; 13; var)	
(3)	PP03		Northeast	110		(114; 12; var)	
(4)	PP04		Northeast	90		(94; 10; var)	
(5)	PP05		Southeast	80		(84; 09; var)	
(6)	PP06		Southeast	90		(94; 10; var)	
(7)	PP07		Southeast	90		(94; 10; var)	
(8)	PP08		Southeast	130		(134; 14; var)	
(9)	PP09		Southeast	150		(154; 16; var)	
(10)	PP10		Southeast	140		(144; 15; var)	
(11)	PP11		South	90		(94; 10; var)	
(12)	PP12		Southwest	90		(94; 10; var)	
(13)	PP13		Southwest	140		(144; 15; var)	
(14)	PP14		Southwest	150		(154; 16; var)	
(15)	PP15		Southwest	140		(144; 15; var)	
(16)	PP16		Northwest	150		(154; 16; var)	
(17)	PP17		Northwest	145		(149; 16; var)	
(18)	PP18		Northwest	115		(119; 13; var)	
(19)	PP19		Northwest	125		(129; 14; var)	
(20)	PP20		Northwest	80		(84; 09; var)	

$$\mathrm{Sy}_{\mathrm{SVPS}} \ni \left[(\mathrm{PP})_{(i,i\le n)}\right], \quad \text{where } i \text{ is a integer and } i \le n = 20 \tag{1}$$

$$\Longrightarrow (\mathrm{PL})_e = \left[(\mathrm{PL})_{(e,e\le k)}\right], \quad \text{where } e \text{ is a integer and } e \le k = 150 \tag{2}$$

$$f(X) = \left[(\mathrm{PL})_{(e,e\le k)}\right] \in \sum_{t=1}^{l} (x)_t, \quad t = [1, 2, 3, \ldots, l] \le k \tag{3}$$

$$\Longrightarrow f(X) \cdot (\mathrm{PL})_e. \tag{4}$$

SVPS system, $\mathrm{Sy}_{\mathrm{SVPS}}$, encompasses distinct parking lots $(\mathrm{PL})_e$ in designed parking points $(\mathrm{PP})_i$, but each parking point PP has occupied $(\mathrm{PL})_e \le k$, the total number of parking lots $(\mathrm{PL})_e$ at each parking point PP is computed in (3) and (4). $f(X) \cdot (\mathrm{PL})_e$, where $f(X)$ aggregates the value of $(x)_t \le l$ upon each computation from $(\mathrm{PL})_e \le k$, depicted in Table 1 including overall required parameters detail. Further, a function is defined $f(Y) = f(X) \cdot (\mathrm{PL})_e$ in order to compute the number of nonvacant NV, that is, $(e, t) = \emptyset$, vacant V, that is, $(e, t) \le (k, l)$, and parking lots $(\mathrm{PL})_e$ by internal system computations.

$$f(Y) = \begin{cases} f(X) \cdot (\mathrm{PL})_e, & (e,t) \le (k,l) \\ f(X) \cdot (\mathrm{PL})_e, & (e,t) = \emptyset \end{cases} \tag{5}$$

The information in Table 1 shows the general-purpose analyzed information used during designing phase of parking points, as part of SVPS system. As shown, for each parking point PP, number of parking lots $(\mathrm{PL})_e$ and the nodes (or sensor nodes) employed to carry the information are listed, including other important details. However, the number of anchor nodes employed in network setup of each parking point is variable (Var) in size, and also the direction mentioned for each parking point PP is measured as an closed estimation (not exact estimation).

Moreover, in Figure 3, a graphical representation is made to show the number of parking point $(\mathrm{PP})_i$ distribution at the various selective locations and with the appropriate directions, over the SVPS-Map positions.

4.1. Network Configuration. In SVPS system, each parking point PP $\in (\mathrm{PP})_i$ is completely designed and networked under the considerations of efficient computing paradigms; thus, it is assumed that all the slave-sensor nodes (SS nodes) are installed at the optimal positions where they could communicate with the master-sensor node (MS node), and further, to local system controller (LSC). In parking points $(\mathrm{PP})_i$, as mentioned above, the available parking lots $(\mathrm{PL})_e$ sizes are not fixed but the standard size used is about 8 feet by 16 feet (width∗length). Each parking lot (PL), in $(\mathrm{PP})_i$, is equipped with slave-sensor node (SS node) or RFID reader installed at the ground position with the protection of solid shield; the ground position is exactly at the intermediate middle position of the PL entrance. Therefore, the registered system's vehicle with the embedded RFID tag should communicate with the SS node installed, using RFID air-interface protocol.

In Figure 4, the identification of vehicle at the parking lot is made through RFID tag that fixed the vehicle's number plate, under the vehicle's front-bonnet; thereby the vehicle

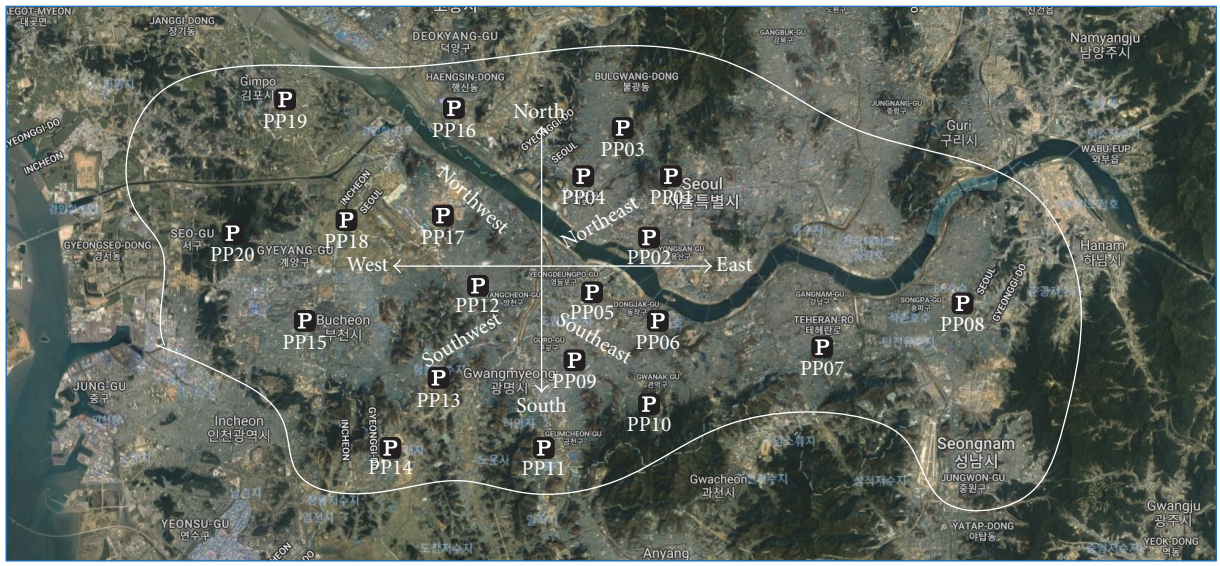

Figure 3: Graphically distribution of parking points.

Figure 4: Smart parking RFID tag and installation.

gets reliable access while entering into the parking lot (PL) assigned. The installation or the fixing process of RFID tag on each vehicle, after registration, is twofold:

(1) In general, the vehicle's number plate is important for the verification purposes in daily life, thus the fixing of RFID tag on vehicle plate will be very useful to get that vehicle identification and verification, at the time vehicle enters into parking point, PP.

(2) As mentioned, each parking lot (PL) is equipped with slave-sensor node installed at the ground position; thus information will be reliably and easily accessible upon vehicle entry at the allocated PL. Through the RFID technology, unlike a barcode, there is not a requirement that RFID tag must be aligned or placed in the line of sight, to the RFID reader installed at each parking lot $(PL)_e$.

For instance, in Figure 5, the number of parking lots $(PL)_e$ is variable sizes in each parking point (PP), but the maximum ten parking lots $(PL)_{(e\leq k=10)}$, that is, $\text{Max}\{1 \leq (PS)_e \leq 10\}$, are assigned to each master-sensor node (MS node), connecting with the wired connection and communication made through the I^2C protocol. RFID technology takes the benefits from WSN technology to increase the converge area, because mainly the RFID systems have been deployed to access the object or the node has RFID tag embedded on, within the short range of about 2 meters (or more). The WSN system efficiently extends the wireless coverage of RFID devices to be reachable to the controller, for example, while transmitting information from sensor nodes to the local system controller in our case. In short, the master-sensor nodes (MS nodes) are directly configured and connected with the slave-sensor nodes installed in parking lots, performing the information read/write functions, and are responsible for transmitting the observed information, through anchor nodes, to local LSC residing in each parking point (PP). Further, MS nodes are responsible for performing controlling operations, through connected slave-sensor nodes (SS nodes), instructed from the CC system. In each parking point, the employed MS nodes are connecting with each other, so each MS node has fully awareness about its neighboring MS node. For controlling and monitoring, the CC system is always responsible; the used RFID technology is flexible and programmable in its usage, according to the commands and instructions. Therefore, the networked system is efficiently regulated through the commands of CC system.

For example, in Table 1, the parking point $(PP)_{(i=12)}$, or PP12, occupied 90 parking lots $(PL)_{(e=90)}$. Thus processing through CC system's database, as a result, few parking lots $(PL)_e$ are vacant (V), that is, $V \in (PL)_e$, such as SVPS-L006, SVPS-L008, SVPS-L010, SVPS-L44, SVPS-L55, and SVPS-L74, and further suppose that rest of them are nonvacant (NV), that is, $NV \in (PL)_e$. From CC system, suppose that,

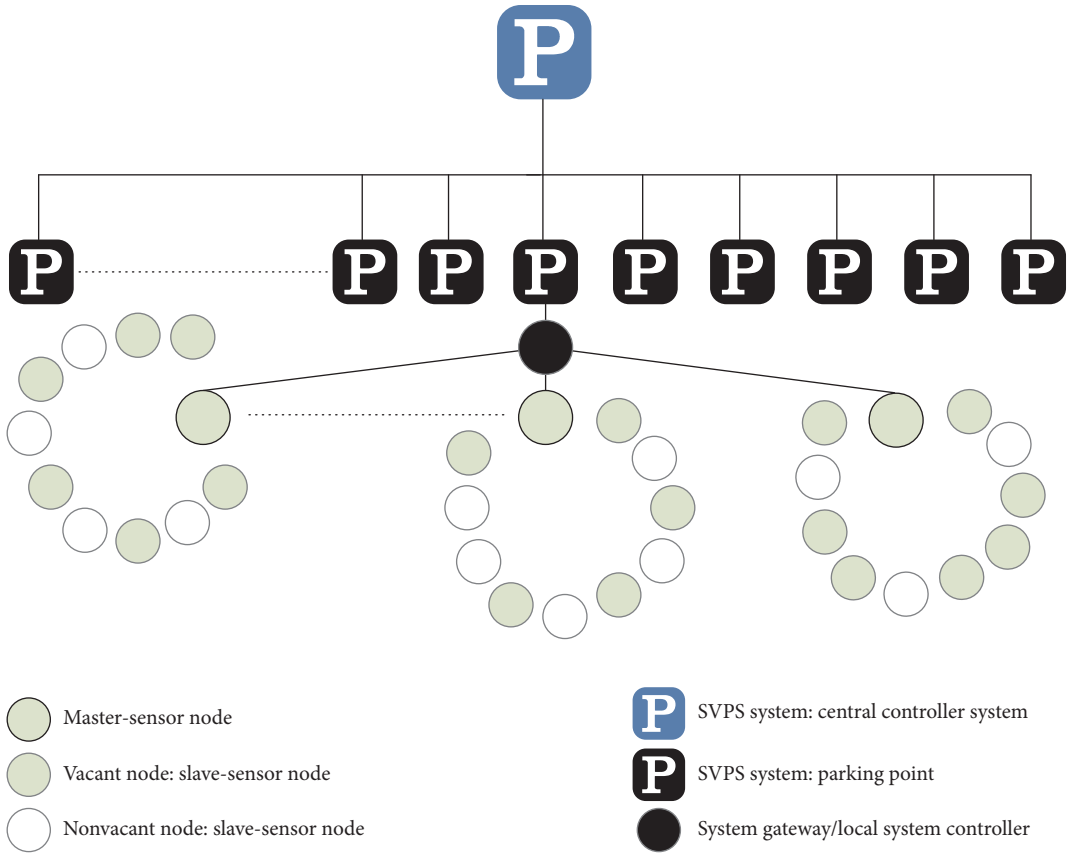

Figure 5: SVPS system: sensor nodes connectivity.

based on the reservation records, the following information is collected:

(1) The six vehicles having vehicle registration IDs $U_{\text{Reg}}, U_{\text{Reg}} \in \text{Sy}_U \in \text{Sy}_{\text{SVPS}}$, such as P561S, T782E, TT89D, TS90P, PK61L, and PK61L.

(2) Registered tag numbers $U_{\text{Tag}}, U_{\text{Tag}} \in U_{\text{Reg}} \in \text{Sy}_U \in \text{Sy}_{\text{SVPS}}$, like IS0004641V780012104P561S, IS0205841P707012104S535S, and so on, illustrated in Figure 5.

(3) Registered users IDs, U_{ID}, $(U_{\text{Tag}}, U_{\text{Reg}}) \in U_{\text{ID}} \in \text{Sy}_U \in \text{Sy}_{\text{SVPS}}$, like Tommy-IS0004641V780012104P561S, Alice-IS0205841P707012104S535S, and so on.

(4) Allocated parking lots $(\text{PL})_e$, that is, $\text{V} \in (\text{PL})_e$, such as PP12-PL065, PP01-PL061, PP01-PL025, PP01-PL022, PP01-PL001, and PP01-PL006.

For each SVPS system's user (U) or $\text{Sy}_U \in \text{Sy}_{\text{SVPS}}$, the above-mentioned detail is compulsory to be registered, a requirement for reservation. Therefore, the information analyzed above, accessing CC system, only those vehicles (e.g., vehicles P561S–PK61L) could get access at the entry gate and onto the designated parking lots (PP12-PL-065–PP01-PL-006) in selective parking point $(\text{PP})_{(i=12)}$. The installed slave-sensors nodes are smart and programmable and fully controlled under the commands given by the CC system. As a result, only that vehicle makes entry onto the selective parking lot of parking point commanded by the CC system. In case, the vehicle enters into the correct parking point, through the reading of sensors installed at the main barrier, mistakenly, the same vehicle will try to make parking onto the parking lot that is not designated for it. Then, upon identification, the installed slave-sensor node will not allow for parking and together generate the indication with red light, a sign that shows the unauthorized parking due to the vehicle identification failure.

For security and ensuring of each registered user (or registered vehicle) under SVPS system, in Figure 6, the main entrance (of parking point) is equipped with two sensors, only at the main entrance. However, the exit gate or exit barrier has one sensor installed to acquire the reading during exit. Thus, at the entrance to the selective parking point (that authorized by the SVPS system), two sensors installed will be active for the vehicle's identification, operating through the follwoing:

(1) The first sensor is installed at the parking entry gate; similar position as parking lot has, at the center of entrance, one step backward of parking boom barrier.

(2) At the same time and the place exactly at the left side, another sensor device is fixed and activated on the panel, which inclusively makes identification of each vehicle at entry point through only with smart RFID card provided by the SVPS system.

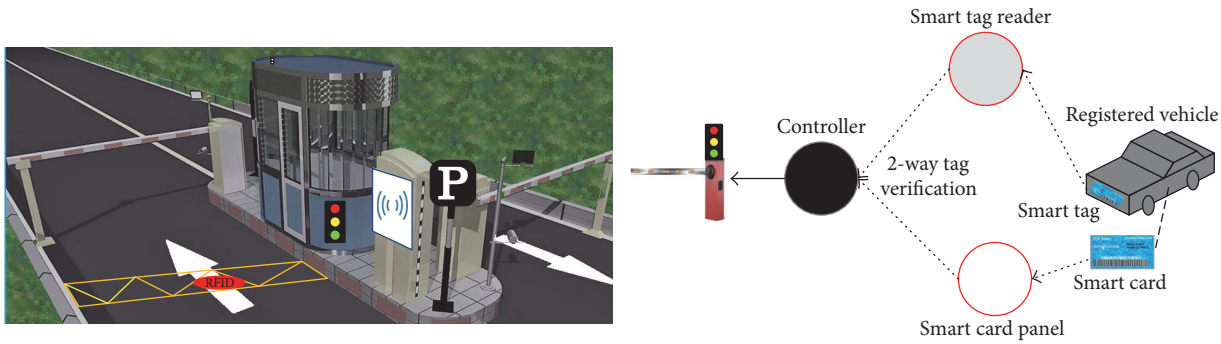

Figure 6: Nodes identification and verification.

5. System Implementation

For parking reservation or request for parking (RFP), users are allowed to access the service online through accessing the SVPS system which keeps the record, in real time, of both nonvacant NV and vacant V parking lots from the parking points, such that $f(Y) = \{f(X) \cdot (\mathrm{PL})_e \mid \mathrm{NV}; \mathrm{V}\} \in (\mathrm{PP})_i \in \mathrm{Sy}_{\mathrm{SVPS}}$; for that, the information is entirely computed from master-slave-sensor nodes (M2S nodes). Therefore, the system users Sy_U (i.e., $\mathrm{Sy}_U \in \mathrm{Sy}_{\mathrm{SVPS}}$) are ensured for their reserved parking lots $(\mathrm{PL})_e$ before arriving at the selective distinct parking points $(\mathrm{PP})_i$. However, some time limitations, but very flexible, are also mentioned and restricted from the SVPS system which enables the session for the user to be reached and used his/her reserved parking lot; further, while the session expires, the SVPS dynamically allocates that parking lot (e.g., the occupied PL) to other users waiting inside system reservation queue ($\mathrm{Sy}_{\mathrm{RQ}}$). The SVPS system used an efficient method to manage and regulated all the reservations pending in the system reservation queue; while it is vacant, the parking lots are assigned to the users who are waiting inside the queue; however the priority scheme should be a main consideration.

SVPS system online is only available for the users that are registered under the system, known as authorized users of the system; other entities are not allowed at all or considered as unauthorized entities. The SVPS system web application is designed, in mind, to be easily usable by the system's users. While successfully logging in to the system online, the user will have access to the various services available. Therefore, the user could access all available services enabled for the desired parking searching and reservation purposes and view the corresponding parking payment details inside the account. To be convenient, the interactive inclusive SVPS-Map is available where all the parking points (i.e., 20 parking points) are located, being useful for the users who are planning to do searching for the closer parking points according to the positions where they are. Thus, based on the user selection for parking point, the user is further allowed to see all the available parking lots, for example, nonvacant (NV) and vacant (V). However, from the system regulations, each user is limited to do reserve the parking lot only once at a time, without any time constraints; but the user could allow reserving another parking lot in distinct parking point in case the user lefts or checks out from the current parking lot that he/she occupied. For payment, the system is designed to manage the total payment of parking service, the user parking utilization is computed on an hourly basis and has opted to make payment by several available reliable methods.

5.1. Smart Access and Reservation. During online SVPS system access, suppose a registered user U has valid entity such that $U \in \mathrm{Sy}_{\mathrm{SVPS}}$. Thus, upon entering the valid user identification ID and password PW, that is, $U(\mathrm{ID}; \mathrm{PW}) \in \mathrm{Sy}_{\mathrm{SVPS}}$, the login credentials are authenticated and then user U is permitted to enter into the system. As a newly entry into the system, the user is allowed to use the overall system' services available inside his/her account, for example, especially the parking searching and reservation service. For that, a request for parking (RFP), the system efficiently displays all the available parking points $(\mathrm{PP})_i$, over the geographical positions designated, on the SVPS-Map, illustrated in Figure 3. Over the SVPS-Map, all the parking points are located a distance far from each other, about the distance of kilometers; the parking point distribution is appropriated for the users who could do the parking reservations over the various locations where they are looking for parking.

$$U \in \mathrm{Sy}_U \in \mathrm{Sy}_{\mathrm{SVPS}},$$
$$U(\mathrm{ID}; \mathrm{PW}) \in \mathrm{Sy}_{\mathrm{SVPS}}$$
$$\Longrightarrow f(U) = \begin{cases} 0, & U \in \mathrm{Sy}_{\mathrm{SVPS}} \\ \Phi, & U \notin \mathrm{Sy}_{\mathrm{SVPS}} \end{cases} \tag{6}$$

$f(U)$ is a function defined to authenticate the user $U(\mathrm{ID}; \mathrm{PW})$, at the time of login to SVPS system online. In case system authenticates the user U, that is, $U(\mathrm{ID}; \mathrm{PW}) = 0$, then user is allowed to use the service RFP for any parking point $\mathrm{PP} \in (\mathrm{PP})_i$,

$$\exists f(U) = 0 \Longrightarrow \left[(\mathrm{PP})_{(i, i \le n)}\right], \quad i = [1, 2, 3, \ldots, n]. \tag{7}$$

Thus, all the available parking points $(\mathrm{PP})_i$ will be displayed to the user, prevalent over various positions onto the SVPS-Map; here user has two options to do searching for the closer parking points $(\mathrm{PP})_i$ and then do a request for parking, which are as follows:

(1) Selective smart parking (SSP): user inputs the specific location as a source location and then searching the closer parking point through SVPS-Map displays all the parking points that are nearest to the selective location of the user. However, the system shows the parking points starting from the closer once, computing the distance from the selective location to each parking points $(\mathrm{PP})_i$. For example, in Figure 7, the user inputs a location as his/her target location and requires getting the closest parking points. For that, the system computes the distance between the target position of user and all the available parking points' positions and thus displays the closer parking points visualized in red color, in converge of red cell, in number order of PP18, PP17, PP12, and PP15.

(2) Smart positional parking (SPP): for this, a request for parking (RFP), the system using SVPS-Map gets automatically the current location of the user and then according to target location input, a number of parking points will be displayed starting from the closest once. For example, in Figure 8, the system gets a current position of the vehicle or a user position and computes the distance with all the available parking points' positions. As a result, the system displays the closer parking points, visualized in red color, in converge of the red cell, with the numbers 01 and 02. Thus, it is concluded that the PP01 has a less distance compared with PP02 within converge of the red cell.

In both cases, the input target location and its distance are computed with all the available parking points and according to the computed results the closer parking points one after another will be identified based on the distance difference with and displayed to the user. However, the total distance between all the available parking points is known by the system, while in design and setup phase of SVPS system.

Mathematical Formulation. Suppose that a function $f(d)$ computes the distance d between each of available parking points $(\mathrm{PP})_i$ such that $\mathrm{Sy}_{\mathrm{SVPS}} : f(d) = d[(\mathrm{PP})_i]_{(ab)}$, where a and b are the distance coordinates of each parking point $\mathrm{PP} \in (\mathrm{PP})_i$. As, at instance, the number of parking points $(\mathrm{PP})_i$ are limited to integer n. Thus, the function $f(d)$ computes the distance between parking points $(\mathrm{PP})_{(i,i\leq n)}$ followed by the value $\mathcal{S}$.

$$\mathcal{S} = \left\{(\mathrm{PP})_1, (\mathrm{PP})_2, (\mathrm{PP})_3, \ldots, (\mathrm{PP})_n\right\} = (\mathrm{PP})_{(i,i\leq n)} \Longrightarrow$$

$$f(d) = d\left[\mathcal{S}\right]_{(ab)} = d\begin{bmatrix} \{(\mathrm{PP})_1, (\mathrm{PP})_2\}; \{(\mathrm{PP})_1, (\mathrm{PP})_3\}; \{(\mathrm{PP})_1, (\mathrm{PP})_4\}; \ldots; \{(\mathrm{PP})_1, (\mathrm{PP})_n\} \\ \{(\mathrm{PP})_2, (\mathrm{PP})_3\}; \{(\mathrm{PP})_2, (\mathrm{PP})_4\}; \{(\mathrm{PP})_2, (\mathrm{PP})_5\}; \ldots; \{(\mathrm{PP})_2, (\mathrm{PP})_n\} \\ \{(\mathrm{PP})_3, (\mathrm{PP})_4\}; \{(\mathrm{PP})_3, (\mathrm{PP})_5\}; \{(\mathrm{PP})_3, (\mathrm{PP})_6\}; \ldots; \{(\mathrm{PP})_3, (\mathrm{PP})_n\} \\ \{(\mathrm{PP})_4, (\mathrm{PP})_5\}; \{(\mathrm{PP})_4, (\mathrm{PP})_6\}; \{(\mathrm{PP})_4, (\mathrm{PP})_7\}; \ldots; \{(\mathrm{PP})_4, (\mathrm{PP})_n\} \\ \{(\mathrm{PP})_5, (\mathrm{PP})_6\}; \{(\mathrm{PP})_5, (\mathrm{PP})_7\}; \{(\mathrm{PP})_5, (\mathrm{PP})_8\}; \ldots; \{(\mathrm{PP})_5, (\mathrm{PP})_n\} \\ \{(\mathrm{PP})_6, (\mathrm{PP})_7\}; \{(\mathrm{PP})_6, (\mathrm{PP})_8\}; \{(\mathrm{PP})_6, (\mathrm{PP})_9\}; \ldots; \{(\mathrm{PP})_6, (\mathrm{PP})_n\} \\ \vdots \\ \vdots \\ \vdots \\ \{(\mathrm{PP})_{n-1}, (\mathrm{PP})_n\} \end{bmatrix}_{(ab)}. \tag{8}$$

For example, the distance d between $(\mathrm{PP})_1$ and $(\mathrm{PP})_2$ is computed as

$$d\left[(\mathrm{PP})_1, (\mathrm{PP})_2\right]_{(ab)} = \sqrt{\left[(\mathrm{PP})_1(a) - (\mathrm{PP})_1(b)\right]^2 + \left[(PP)_2(a) - (\mathrm{PP})_2(b)\right]^2}. \tag{9}$$

Further, assume that L_1 and L_2 are the target location and the current location of registered user U input, then the system is mapping the input values $L_1 \mapsto d[\mathcal{S}]_{(ab)}$ and $L_2 \mapsto d[\mathcal{S}]_{(ab)}$ and resulted in the closer parking points $(\mathrm{PP})_i$ visualized in Figure 9.

For the user convenient and based on the location inputs, in general, the system executes the user request and then displays the available parking points $(\mathrm{PP})_i$ in order starting from the closest to that user's input location. However, in Figures 7 and 8, the system only displayed the closer parking points, that is, PP18, PP17, PP12, and PP15 and PP01 and PP02, with an indication of available or vacant parking lots $(\mathrm{PL})_e$. More precisely, in Figure 9, the indication symbol in red and black colors shows the number of vacant V parking lots $(\mathrm{PL})_e$, such as 10 in PP01, 23 in PP02, 44 in PP18, 31 in PP17, and 52 in PP15; however, in case the parking lots $(\mathrm{PL})_e$ are not vacant NV, then they are displayed with zero, for example, in case of PP12.

In Figure 9, using the cases of Figures 7 and 8, the system computed a number of vacant V parking lots $(\mathrm{PL})_e$ indicated individually for the selective parking point $(\mathrm{PP})_i$, but not

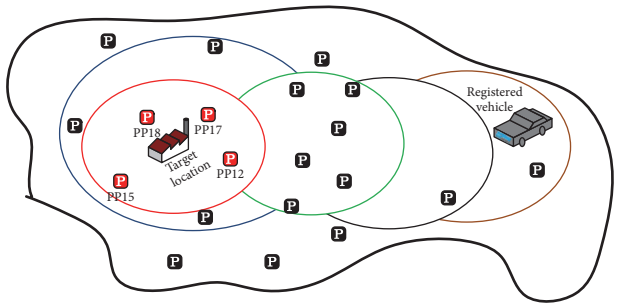

Figure 7: Selective smart parking using SVPS-Map.

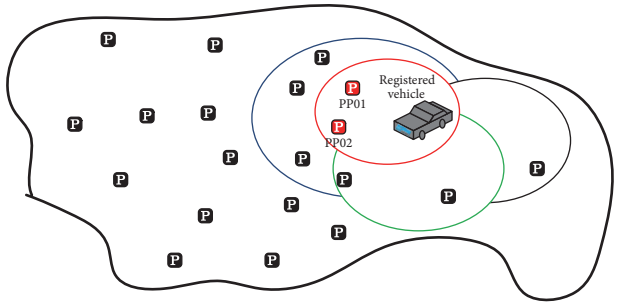

Figure 8: Smart positional parking using SVPS-Map.

for all. Therefore, during request for parking, users save time by knowing about the vacant (or nonvacant) parking lots in advance before they go to do parking reservations. The parking reservation became more straightforward to the users, as they already know the available parking points and the vacant parking lots, to submit the request for parking. The SVPS system employed a general-purpose strategy to design and visualize the number parking lots in parking points, which is useful at the time of parking reservations. In Figure 10, for example, PP01 occupied 120 parking lots in consecutive order and is displayed in the form of fixed size blocks; the vacant parking lots are represented in white color and at the same time, the gray color represented nonvacant parking lots. Thus, through request for parking, user has an option to select any of parking lots among the vacant parking lots. By selecting the specific parking lot, for example, parking lot number PP01-PL010 shown in white color, further system processes a request for parking and allocates it for the user who requested. At the same time, the allocated parking lot PP01-PL010 information is updated in the database, and the corresponding master-slave-sensor nodes are triggered to allow access to the authorized vehicle to do parking (through reading of RFID tag information). Following the user request for parking, that is, fixed time parking and timed parking, the session for parking (to utilize the allocated parking lot) is added, and the master-slave-sensor nodes are very known or have information that allows the vehicle to do parking according to the session mentioned and exit while time finished. However, the user has option to extend the existing declared session of parking using SVPS system online; otherwise, the system regulates the parking timing on an hourly basis, for example, using any time parking feature.

For information acquisition, in short, while user request for parking is confirmed and reserved with the allocation of the parking lot, the system actives the sensor nodes with the user (or vehicle) registered ID using write function; therefore only that user could do the parking onto the parking lot allocated. All the operations, such as read and write operations, fully instruct, operate, and are controlled by the central controller system (CC system), and the corresponding computed information is always simultaneously updated into the system database or SVPS-DB. In Figure 11, SVPS-DB designed (or created) tables show the overall information keep while each time the system gets reading from master-slave-sensor nodes (i.e., slave-sensor nodes) installed at the parking point PP $\in$ $(PP)_i$. Further, some of important operations performed and managed in SVPS-DB are following:

(1) For each parking point PP $\in$ $(PP)_i$, SVPS-DB has fixed number of tables for keeping the information up to date from the master-slave-sensor nodes. However, it depends directly on the number of parking lots $(PL)_e$ being installed with the sensor nodes in each parking point. For example, suppose that, in parking point 03 or PP03, total number of parking lots are 114 installed independently with the sensor nodes; therefore for information acquisition, the starting 110 parking lots are directly connected and managed through 11 master-sensor nodes, that is, 10 slave-nodes/1 master-sensor node, and the remaining 4 parking lots out of 114 are through 12th master-sensor node. This means overall 114 slave-sensor nodes are managed through 12 master-sensor nodes; more details are depicted in Table 1 showing the information of each parking point installed with the fixed number of master-slave-sensor nodes. Thus, in the SVPS-DB, the system maintains 12 tables to keep the information individually from 12 master-sensor nodes designated to control the fixed number of slave-sensor nodes installed to get the reading while vehicle checking in and checking out. However, information collected from master-slave-sensor nodes has also replicated in local system controller (LSC) setup in each parking point.

(2) Further, among other several tables, the system maintains two main tables in SVPS-DB: (1) parking status table and (2) parking for a request or parking reservation table. Parking status table keeps the record/information of parking lots, which are vacant and nonvacant based on the master-slave-sensor nodes reading information, from each parking point. Then, the collected information will be displayed to the users while searching for parking using SVPS system online. At the other side, parking for request table keeps the record of each reservation made by the users and the equivalent information that slave-sensor nodes are fully activated only for registered vehicles having RFID tags.

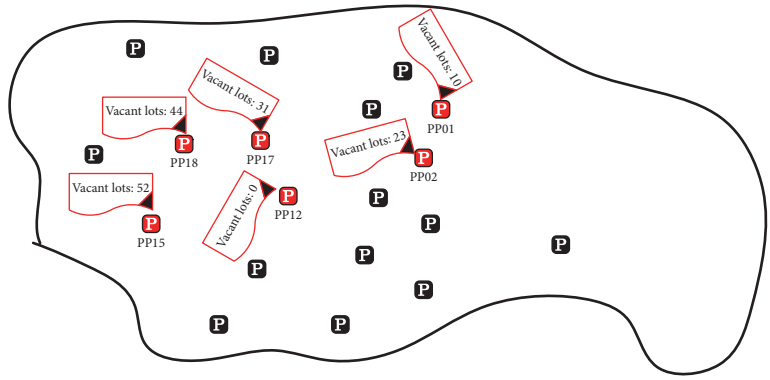

Figure 9: Computation of vacant/nonvacant parking lots using SVPS-Map.

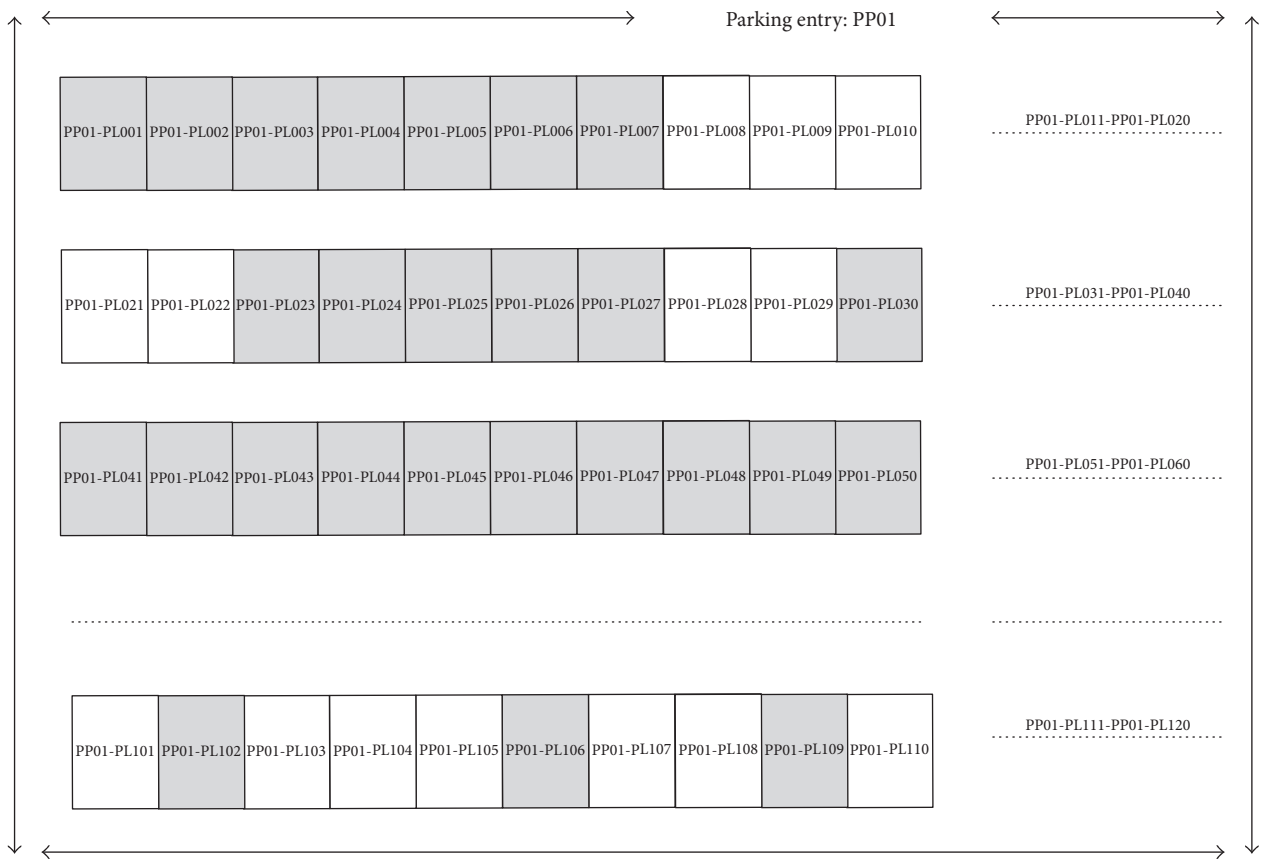

Figure 10: A design strategy for parking reservation.

6. Results and Discussion

At the time of user's login, the SVPS system authenticates the user entered ID and the password. Therefore, after authentication process, the user will be allowed access to use the available system's services; each time the user attempts to access the system services online, a session is activated for a fixed time (e.g., 9 minutes after first successful attempted of login) and can be further extendable. However, the current study or SVPS system is not designed under the consideration of efficient security paradigms in mind and only used the available general-purpose security mechanisms available inside operation system (i.e., firewall and commercial security tools) and conventional SSL and TLS solutions during the web searching. In early future, the study aims to design a fully secured security system that will fulfill all the requirements of security against the potential adversaries, while connecting with and managing the system online via Internet access.

For parking request, the overall system is efficient to manage the number of users in parallel. But in the situation, at same time, two or more users are targeting the same parking lot $PL \in (PL)_e$ in parking point $PP \in (PP)_i$, then the system

Table 2: Write operation and computation.

Function code, write operation: 1111			
Levels	System nodes	Execution	Confirmation, bit set = 1
1	CC system to LSC	1111	1110
2	LSC to MS node	1110	1100
3	MS node to SS node	1100	1000
4	SS node	1000	0000

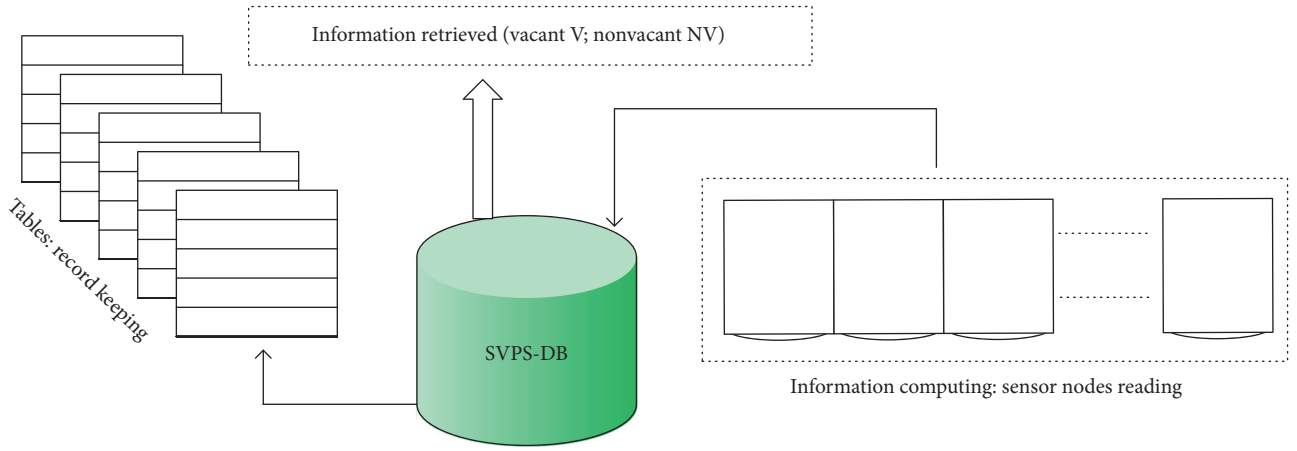

Figure 11: SVPS-DB: information computing and retrieving.

employs the scheduling algorithm, first-come, first-served (FCFS), and nonpreemptive approach, in order to manage the users. Suppose that, in Figure 11, three users input the request, request for parking, for the same parking lot PL01-PL004 in parking point PP01, then the system computes the exact time (TR time of arrival) at which the first user among other has insert the request. Followed by the system, the time computed for user 02 is 02:12:55 (hour/minutes/seconds) which is less compared with the arrival times of user 01 and user 03 as 02:32:21 (hour/minutes/seconds) and 02:39:41 (hour/minutes/seconds). Therefore, a message or exception will be generated "the parking lot you are requesting is not available"; moreover, the other users are allowed any time to reserve the other available parking lots inside the same parking point. Here, in Figure 12, short IDs, such as user 01, user 02, and user 03, are used instead of long IDs defined by the system in Section 4.

The system is efficient in computing of information at each level, that is, sensor nodes to the local system controller to CC system and vice versa, of parking point PP $\in$ $(PP)_i$; therefore the users can get the overall required information in minimal time and in efficient manners. To compute and ensure the information at each level of computation, the general-purpose special functional codes are defined and employed by the system (each function code has 4-bit size) and are as follows: (1) for reading information, the code defined is 0000, and (2) for writing information, the code defined is "1111." These are, for example, four-bit codes defined to perform the read and write functions from/to the master-slave-sensor nodes (M2S nodes). More precisely, the user-defined codes executing with the change of a bit at each level of computation during read and write operations. In the case of request for parking, the CC system performs the write operation to assign and activate the RFID reader or slave-senor node (SS node) installed at the parking lot with vehicle information having RFID tag fixed; thus upon the arrival of vehicle at the allocated parking lot the RFID tag information will read by the SS node. For that, CC system inputs a command "1111" for write operation; then the following execution will be performed at the each level of computation; that is, logically four levels are defined: (1) CC system to LSC, (2) LSC to MS node, (3) MS node to SS node, and (4) SS node. Thus, the write operation is performed as follows by these four levels: with the confirmation, bit set = 1 changing at each level after successful execution.

In Table 2, the original code to write information is 1, or write code = 1, but the additional 3 bits or three 1's are used and exclusive or XOR operation is performed at each level. The bit set = 1 is XOR with the least significant bit of "1111" at each level to get "0000" for confirmation. Therefore, at level 4, the information is written using write code = 1 (i.e., "1000") and with the confirmation of "0000."

Similarly, in Table 3, CC system inputs a read code = 0 to get information or reading from the SS node, for example, whether the parking lot is currently vacant or nonvacant, RFID tag information of the authorized vehicle and so on. For that, the additional three 0's are added to read code which becomes "0000" and for confirmation, a set bit = 0 has been XOR each time with the least significant bit of "0000" to get finally the number of 1's or "1111" as confirmation observed

Table 3: Read operation and computation.

Function code, reading operation: 0000			
Levels	System nodes	Execution	Confirmation, bit set = 0
1	CC system to LSC	0000	0001
2	LSC to MS node	0001	0011
3	MS node to SS node	0011	0111
4	SS node	0111	1111

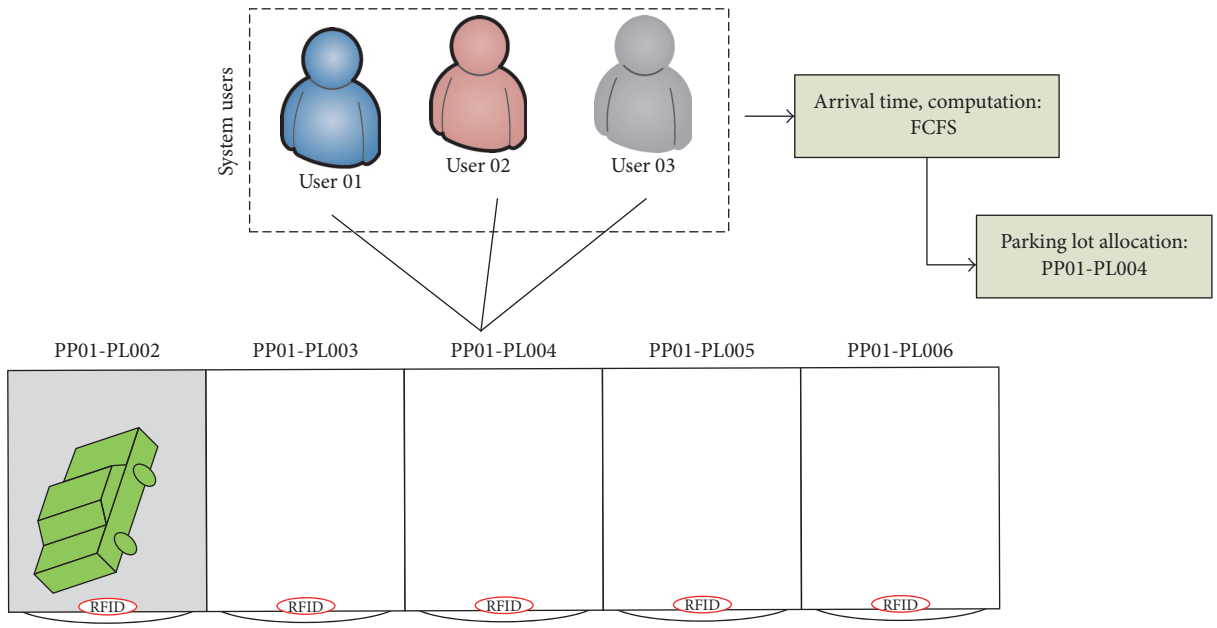

Figure 12: Parking allocation and management.

at each level. Further, the information being collected from RFID tag is intended to transmit back; according to command given by CC system, the sequence of the execution from nodes has been changed, as (1) RFID tag to SS node, (2) SS node to MS node, (3) MS node to LSC, and (4) LSC to CC system.

In Figure 13, the given information is the information transmitted from the parking point to the CC system. As mentioned, each parking point PP $\in$ $(PP)_i$ has been equipped or set up with various nodes to get instructions or commands and transmit reading information, from/to the CC system. For reading information, for example, the slave-sensor node gets reading from the RFID tag, and if the computed value is "0000" starting from the most significant bit or 0 bits which shows that the parking lot status is currently nonvacant, then confirmation set bit = 0 is applied to get the value of "0001." Continuously, at each level, set bit = 0 is XOR in order to get the final confirmation at level 4 as "1111" and execution value "0111." Similarly, in Figure 14, slave-sensor node sets the value as "1111," meaning that the parking lot is currently vacant because the most significant bit is "1" and remaining three bits or 1's are added in order to get the confirmation at the next levels by applying set bit = 1. Thus, at level 4, CC system gets "1000" and with the confirmation "0000." Here, in both cases, whether parking lot is vacant or nonvacant, the computed values are the same as the values when CC system sends commands for write and read operations. Notably, each time command sent from CC system or sensor nodes replied the information back to CC system, the distinct identifications from them are also added along with the information. Thus, there is no conflict at all while command is transmitted or information is replied.

As a consequence, each piece of time information has been collected and transmitted back to the CC system and simultaneously kept storage into the SVPS-DB; therefore, the received information from the sensor nodes has been analyzed to acquire the number of parking lots whether being vacant or nonvacant individually from each parking point. While the slave-sensor node senses the vehicle tag information and upon verification, the vehicle will allow doing parking onto the parking lot, at the same time CC system gets the same information and updates the status of that parking lot as nonvacant. At the other side, upon the session completion assigned for the vehicle to utilize the parking lot, the vehicle has to be an exit from the parking lot; thus upon exit, the system updates the status of the parking lot as vacant for other users. However, in case the user wants to extend the parking session of same parking lot using SVPS system online, system keeps the parking status vacant till the vehicle exits. In Table 4, the numerous vacant

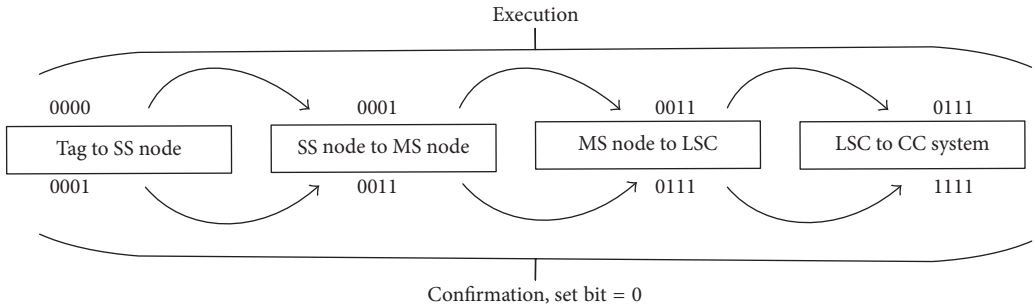

Figure 13: Nonvacant parking lot: reading information flow.

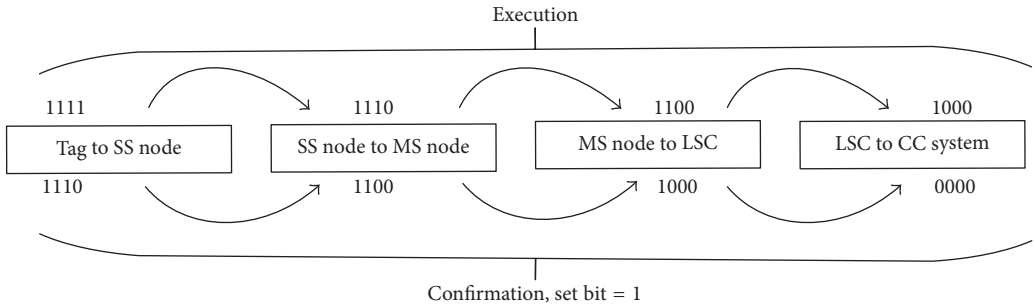

Figure 14: Vacant parking lot: reading information flow.

and nonvacant parking lots information is displayed which is measured individually from the parking points through the sensor nodes.

In Table 4, the randomly six parking points using Figure 9, that is, PP01, PP02, PP18, PP17, PP12, and PP15 $\in (PP)_i$, are selected among others and the status for the numerous parking lots is observed. However, the numerous parking lots are also randomly chosen from the selective six parking points. Thus, the status whether the parking lots are vacant = 1 or nonvacant = 0 in parking points is observed directly from the sensor nodes installed. For the system users, this information is very important, with which they are able to make parking reservations. Upon user selection for parking point, the information simultaneously displays the status of the parking lots; therefore the user has awareness about the parking lots status and, of course, saves time that might be wasted during searching for parking lots' availability.

As mentioned, the SVPS system is efficient in its overall processing, such as in parking searching and reservations and others; thereby the numerous of parking lots in parking points $(PP)_{(i \le k)}$ are equipped with the sensor nodes. The networked master-slave-sensor nodes are entirely controlled and managed through the CC system, in cases of command or information write operation and read operation, and are fully functional to get or transmit information. As soon, the slave-sensor node gets the vehicle information through its embedded RFID tag, the CC system will also get the same computed information in minimal time session. However, the time always varies in cases where the CC system gets/reads information and instructs/writes command, from/to the sensor nodes. For each transaction, that is, write and read, the total time that required completing the process

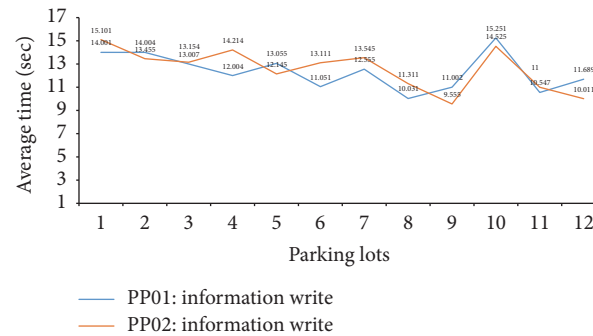

Figure 15: Average time computation during information writing.

is efficiently observed by the system. For that, in Figures 15–18, few successful experiments are conducted to compute the total time required during performing the write and read operations. The computed time in each experiment is average computed time in seconds and the overall computed results are considered as the most appropriate measurements according to our best knowledge. Further, the measurements are conducted using simulation, without any real equipment used, running in a computer machine Intel Core i7 system. Therefore, overall results conducted are nearly assumed to be average and accurate results. The average times in Figures 15–18 are measured by CC system input commands many times, and for these measurements, simulation continuously run about 5880 seconds.

In Figures 15 and 16, 12 experiments are randomly selected and performed to get the total average time while the CC system sent information (i.e., write commands) to and got

Table 4: Reading operation and computation.

Parking lots status (vacant = 1; nonvacant = 0)										
PP01		PP02		PP18		PP17		PP12		PP15	
LP001	0	LP001	1	LP001	0	LP001	1	LP001	0	LP001	1
LP002	0	LP002	0	LP002	1	LP002	1	LP002	0	LP002	0
LP003	0	LP003	0	LP003	1	LP003	1	LP003	0	LP003	0
LP004	1	LP004	1	LP004	1	LP004	1	LP004	0	LP004	1
LP005	0	LP005	0	LP005	1	LP005	1	LP005	0	LP005	1
LP006	0	LP006	1	LP006	1	LP006	1	LP006	0	LP006	1
LP007	1	LP007	0	LP007	1	LP007	1	LP007	0	LP007	1
LP008	1	LP008	0	LP008	1	LP008	1	LP008	0	LP008	1
LP009	1	LP009	0	LP009	0	LP009	1	LP009	0	LP009	1
LP010	0	LP010	1	LP010	1	LP010	1	LP010	0	LP010	1
LP011	1	LP011	1	LP011	1	LP011	1	LP011	0	LP011	1
LP012	0	LP012	1	LP012	1	LP012	1	LP012	0	LP012	1
LP013	1	LP013	1	LP013	1	LP013	1	LP013	0	LP013	1
LP014	0	LP014	1	LP014	1	LP014	1	LP014	0	LP014	1
LP015	0	LP015	1	LP015	1	LP015	1	LP015	0	LP015	1
LP016	0	LP016	1	LP016	1	LP016	1	LP016	0	LP016	1
LP017	1	LP017	1	LP017	1	LP017	1	LP017	0	LP017	1
LP018	0	LP018	0	LP018	1	LP018	1	LP018	0	LP018	1
LP019	0	LP019	1	LP019	1	LP019	1	LP019	0	LP019	1
LP020	0	LP020	1	LP020	1	LP020	1	LP020	0	LP020	1
LP021	0	LP021	1	LP021	0	LP021	1	LP021	0	LP021	1
LP022	0	LP022	1	LP022	1	LP022	1	LP022	0	LP022	1
LP023	1	LP023	1	LP023	1	LP023	1	LP023	0	LP023	1
LP024	0	LP024	1	LP024	0	LP024	1	LP024	0	LP024	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
LP120	0	LP120	1	LP115	0	LP145	1	LP090	0	LP140	1

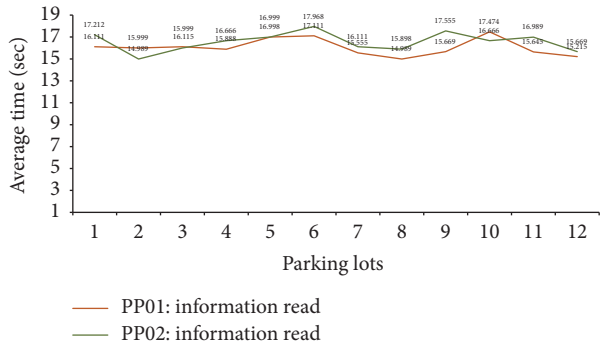

Figure 16: Average time computation during information reading.

a response (i.e., read commands) from the sensor nodes networked in the parking points PP01 and PP02. Similarly, in Figures 17 and 18, the total average time is observed in 12 experiments, individually conducted, while information is sent to and received from sensor nodes networked in the parking points PP18, PP17, PP12, and PP15.

7. Conclusion and Future Work

The proposed study or SVPS system has been contributed to developing a more advanced and smart parking system, and to overcome the problematics of vehicle parking (during parking searching) in a metropolitan city mostly today's parking systems have. For the smart parking, SVPS system has been contributed to fulfilling several objectives, that is, parking searching and reservations, which are the key requirements of today's smart parking system. In spite of the other services provided, the parking searching and reservation module is one of the important modules which provides efficient ways to do searching for the parking points using SVPS-Map facility available online through login into the accounts provided by the system for the registered users. Therefore, users are able to do searching for the parking points, according to their interest, having the indications of vacant or nonvacant parking lots to perform further parking reservations. As a result, users could save enough time that is mostly wasted in searching for parking and the parking lots availability prior to the final reservations. For information computation, the SVPS system

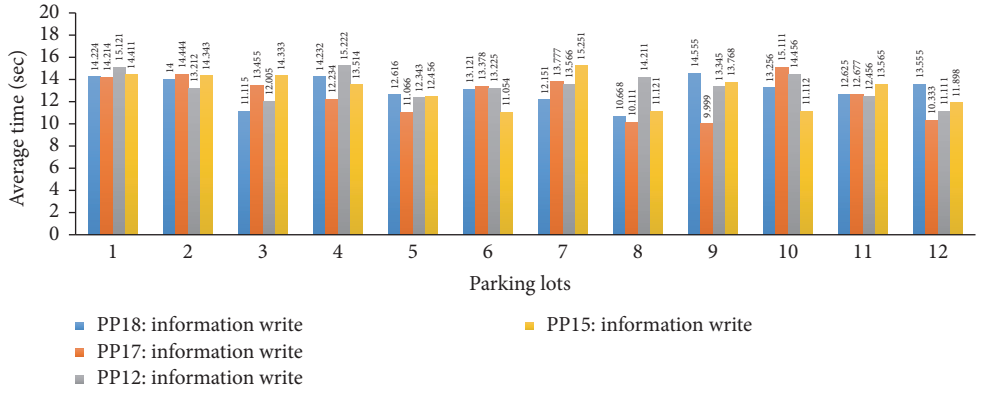

Figure 17: Selective parking points and information writing.

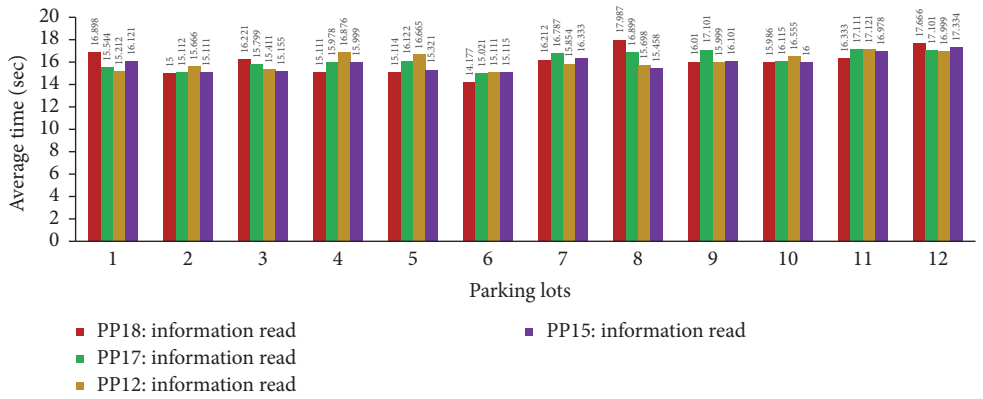

Figure 18: Selective parking points and information reading.

employed advanced integrated RFID and WSN or RFID-WSN technology to network all its parking points which are fully set up with the sensor nodes; therefore each time the RFID tags read the information, for example, vehicle check-in and checkout information, transmitted to the CC system. Overall the information either read or write, the instructions, or commands are always inputted by the CC system.

This study is rigorous in its current development stages and for the further future enhancements, the current study aims at the vehicle parking reservations, resource management, and remote monitoring. In future, therefore the important modules, such as (1) indoor parking interactive mapping, (2) the external and internal system security design, and (3) secure-application access for smart cellular devices, will be designed and implemented as parts of SVPS system.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Faculty Research Fund of Sejong University in year 2017, Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (no. 2017-0-00756, Development of Interoperability and Management Technology of IoT System with Heterogeneous ID Mechanism), Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2014M3C4A7030503), and the National Research Foundation of Korea (NRF) granted by the Korea Government (MSP) (no. 2017R1A2B4006667).

References

[1] T. N. Pham, M.-F. Tsai, D. B. Nguyen, C.-R. Dow, and D.-J. Deng, "A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies," *IEEE Access*, vol. 3, pp. 1581–1591, 2015.

[2] A. Khanna and R. Anand, "IoT based smart parking system," in *Proceedings of the 2016 International Conference on Internet of Things and Applications, IOTA 2016*, pp. 266–270, Pune, India, June 2016.

[3] C. Rhodes, W. Blewitt, C. Sharp, G. Ushaw, and G. Morgan, "Smart routing: A novel application of collaborative path-finding to smart parking systems," in *Proceedings of the 16th IEEE Conference on Business Informatics, CBI 2014*, pp. 119–126, Geneva, Switzerland, July 2014.

[4] N. Mejri, M. Ayari, R. Langar, F. Kamoun, G. Pujolle, and L. Saidane, "Cooperation versus competition towards an efficient

parking assignment solution," in *Proceedings of the 2014 1st IEEE International Conference on Communications, ICC 2014*, pp. 2915–2920, Sydney, Australia, June 2014.

[5] C. J. Rodier and S. A. Shaheen, "Transit-based smart parking: An evaluation of the San Francisco Bay area field test," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 2, pp. 225–233, 2010.

[6] A. Bagula, L. Castelli, and M. Zennaro, "On the design of smart parking networks in the smart cities: An optimal sensor placement model," *Sensors*, vol. 15, no. 7, pp. 15443–15467, 2015.

[7] Z. Pala and N. Inanç, "Smart parking applications using RFID technology," in *Proceedings of the 1st Annual RFID Eurasia*, pp. 1–3, Istanbul, Turkey, September 2007.

[8] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.

[9] J. Majrouhi Sardroud, "Influence of RFID technology on automated management of construction materials and components," *Scientia Iranica*, vol. 19, no. 3, pp. 381–392, 2012.

[10] L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and R. Vergallo, "Integration of RFID and WSN technologies in a Smart Parking System," in *Proceedings of the 22nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2014*, pp. 104–110, Split, Croatia, September 2014.

[11] Pala, Zeydin, and N. Inanc, "Utilizing RFID for smart parking applications," *Facta universitatis-series: Mechanical Engineering*, vol. 7, no. 1, pp. 101–118, 2009.

[12] K. Hassoune, W. Dachry, F. Moutaouakkil, and H. Medromi, "Smart parking systems: A survey," in *Proceedings of the 11th International Conference on Intelligent Systems: Theories and Applications, SITA 2016*, pp. 1–6, Mohammedia, Morocco, October 2016.

[13] C. W. Hsu, M. H. Shih, H. Y. Huang, Y. C. Shiue, and S. C. Huang, "Verification of smart guiding system to search for parking space via DSRC communication," in *Proceedings of the 2012 12th International Conference on ITS Telecommunications, ITST 2012*, Taipei, Taiwan, November 2012.

[14] M. Du, J. Fang, and H. Cao, "A new solution for city parking guiding based on Internet of Things and multi-level multi-agent," in *Proceedings of the 2011 International Conference on Electronics, Communications and Control, ICECC 2011*, pp. 4093–4096, Ningbo, China, September 2011.

[15] L. Lambrinos and A. Dosis, "DisAssist: An internet of things and mobile communications platform for disabled parking space management," in *Proceedings of the 2013 IEEE Global Communications Conference, GLOBECOM 2013*, pp. 2810–2815, Atlanta, GA, USA, December 2013.

[16] H. Wang and W. He, "A reservation-based smart parking system," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, Shanghai, China, April 2011.

[17] Y. Geng and C. G. Cassandras, "A new "Smart Parking" System Infrastructure and Implementation," *Social and Behavioral Sciences*, vol. 54, pp. 1278–1287, 2012.

[18] S. Qin and X. Yao, "An intelligent parking system based on GSM module," *Applied Mathematics & Information Sciences*, vol. 7, no. 1L, pp. 55–59, 2013.

[19] R. E. Barone, T. Giuffre, S. M. Siniscalchi, M. A. Morgano, and G. Tesoriere, "Architecture for parking management in smart cities," *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 445–452, August 2014.

[20] S. A. Faheem, G. M. Mahmud, M. Khan, H. Rahman, and Zafar, "A Survey of Intelligent Car Parking System," *Journal of Applied Research and Technology*, vol. 11, no. 5, pp. 714–726, 2013.

[21] F. I. Shaikh, P. N. Jadhav, S. P. Bandarkar et al., "Smart Parking System Based on Embedded System and Sensor Network," *International Journal of Computer Applications*, vol. 140, no. 12, 2016.

[22] A. Correa, G. Boquet, A. Morell, and etal., "Autonomous car parking system through a cooperative vehicular positioning network," *Sensors*, vol. 17, no. 4, article no. 848, 2017.

[23] S. Cui, M. Wu, C. Liu, and N. Rong, "The research and implement of the intelligent parking reservation management system based on ZigBee technology," in *Proceedings of the 2014 6th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2014*, pp. 741–744, Zhangjiajie, China, January 2014.

[24] D. J. Bonde, R. S. Shende, A. S. Kedari, K. S. Gaikwad, and A. U. Bhokre, "Automated car parking system commanded by Android application," in *Proceedings of the 4th International Conference on Computer Communication and Informatics (ICCCI '14)*, pp. 1–4, IEEE, Coimbatore, India, January 2014.

[25] M. O. Reza et al., "Smart parking system with image processing facility," *International Journal of Intelligent Systems and Applications*, vol. 4, no. 3, pp. 41–47, 2012.

[26] M. Y. I. Idris, E. M. Tamil, Z. Razak, N. M. Noor, and L. W. Km, "Smart parking system using image processing techniques in wireless sensor network environment," *Information Technology Journal*, vol. 8, no. 2, pp. 114–127, 2009.

[27] S. Banerjee, P. Choudekar, and M. K. Muju, "Real time car parking system using image processing," in *Proceedings of the 2011 3rd International Conference on Electronics Computer Technology, ICECT 2011*, pp. 99–103, Kanyakumari, India, April 2011.

[28] I. Song, K. Gowan, J. Nery et al., "Intelligent parking system design using FPGA," in *Proceedings of the 2006 International Conference on Field Programmable Logic and Applications, FPL*, pp. 1–6, Madrid, Spain, August 2006.

[29] M. Gallo, L. D'Acierno, and B. Montella, "A multilayer model to simulate cruising for parking in urban areas," *Transport Policy*, vol. 18, no. 5, pp. 735–744, 2011.

[30] D. Van der Goot, "A model to describe the choice of parking places," *Transportation Research Part A: General*, vol. 16, no. 2, pp. 109–115, 1982.

[31] R. G. Thompson and A. J. Richardson, "A parking search model," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 3, pp. 159–170, 1998.

[32] W. Young and M. Taylor, "A parking model hierarchy," *Transportation*, vol. 18, no. 1, pp. 37–58, 1991.

[33] M. M. Berenger Vianna, L. da Silva Portugal, and R. Balassiano, "Intelligent transportation systems and parking management: Implementation potential in a Brazilian city," *Cities*, vol. 21, no. 2, pp. 137–148, 2004.

[34] J. Li, H. Hu, Q. Ke, and N. Xiong, "A novel topology link-controlling approach for active defense of a node in a network," *Sensors*, vol. 17, no. 3, article no. 553, 2017.

[35] P.-F. Wu, F. Xiao, C. Sha, H.-P. Huang, R.-C. Wang, and N.-X. Xiong, "Node scheduling strategies for achieving full-view area coverage in camera sensor networks," *Sensors*, vol. 17, no. 6, article no. 1303, 2017.

[36] H. Cheng, Y. Chen, N. Xiong, and F. Li, "Layer-based data aggregation and performance analysis in wireless sensor networks,"

Journal of Applied Mathematics, vol. 2013, Article ID 502381, 2013.

[37] F. Xia, R. Hao, J. Li, N. Xiong, L. T. Yang, and Y. Zhang, "Adaptive GTS allocation in IEEE 802.15.4 for real-time wireless sensor networks," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1231–1242, 2013.

[38] Roussos and George, *Networked RFID: systems, software and services*, Springer Science & Business Media, 2008.

[39] J. P. Benson, T. O'Donovan, P. O'Sullivan et al., "Car-park management using wireless sensor networks," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks, LCN 2006*, pp. 588–595, Tampa, FL, USA, November 2006.

[40] Shaheen and Susan, *Smart parking management field test: A bay area rapid transit (bart) district parking demonstration*, Institute of Transportation Studies, 2005.

[41] R. Yusnita, F. Norbaya, and N. Basharuddin, "Intelligent Parking Space Detection System Based on Image Processing," *International Journal of Innovation, Management and Technology*, vol. 3, article no. 232, no. 3, 2012.

[42] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[43] M. M. Rashid, A. Musa, M. A. Rahman, N. Farahana, and A. Farhana, "Automatic parking management system and parking fee collection based on number plate recognition," *International Journal of Machine Learning and Computing*, vol. 2, no. 2, pp. 93–98, 2012.

[44] H. R. H. Al-Absi, J. D. D. Devaraj, P. Sebastian, and Y. V. Voon, "Vision-based automated parking system," in *Proceedings of the 10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010*, pp. 757–760, Kuala Lumpur, Malaysia, May 2010.

[45] E. Karbab, D. Djenouri, S. Boulkaboul, and A. Bagula, "Car park management with networked wireless sensors and active RFID," in *Proceedings of the IEEE International Conference on Electro/Information Technology (EIT '15)*, pp. 373–378, IEEE, Dekalb, Ga, USA, May 2015.

[46] E. P. C. EPCglobal, "Radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz–960 mhz version 1.0. 9," 2004, K. Chiew et al./On False Authenticationsfor C1G2 Passive RFID Tags 65.

[47] C. Ying and Z. Fu-Hong, "A system design for UHF RFID reader," in *Proceedings of the 2008 11th IEEE International Conference on Communication Technology, ICCT 2008*, pp. 301–304, Hangzhou, China, November 2008.

[48] A. Shahzad, M. Lee, H. D. Kim, S.-M. Woo, and N. Xiong, "New security development and trends to secure the SCADA sensors automated transmission during critical sessions," *Symmetry*, vol. 7, no. 4, pp. 1945–1980, 2015.

[49] A. Shahzad, M. Lee, C. Lee et al., "The protocol design and New approach for SCADA security enhancement during sensors broadcasting system," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14641–14668, 2016.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 6897523, 9 pages
https://doi.org/10.1155/2018/6897523

Research Article

Data Processing Delay Optimization in Mobile Edge Computing

Guangshun Li, Jiping Wang, Junhua Wu, and Jianrong Song

School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

Correspondence should be addressed to Guangshun Li; 30752585@qq.com

Received 13 October 2017; Accepted 9 January 2018; Published 6 February 2018

Academic Editor: Shangguang Wang

With the development of Internet of Things (IoT), the number of mobile terminal devices is increasing rapidly. Because of high transmission delay and limited bandwidth, in this paper, we propose a novel three-layer network architecture model which combines cloud computing and edge computing (abbreviated as CENAM). In edge computing layer, we propose a computational scheme of mutual cooperation between the edge devices and use the Kruskal algorithm to compute the minimum spanning tree of weighted undirected graph consisting of edge nodes, so as to reduce the communication delay between them. Then we divide and assign the tasks based on the constrained optimization problem and solve the computation delay of edge nodes by using the Lagrange multiplier method. In cloud computing layer, we focus on the balanced transmission method to solve the data transmission delay from edge devices to cloud servers and obtain an optimal allocation matrix, which reduces the data communication delay. Finally, according to the characteristics of cloud servers, we solve the computation delay of cloud computing layer. Simulation shows that the CENAM has better performance in data processing delay than traditional cloud computing.

1. Introduction

Data storage and computation are two critical problems in cloud computing, which provides a method to solve the limited storage and computing speed of computers or mobile phones [1]. With the development of Internet of Things (IoT), the amount of data transmission shows a trend of exponential increment [2, 3]. It is predicted that the growth trend of data traffic would be eightfold between 2014 and 2020, which will bring a huge challenge for cloud computing [4, 5]. On one hand, limited bandwidth has adverse effects on the efficiency of data transmission. On the other hand, terminal is usually far from the cloud servers and data transmission with long distance increases the transmission delay, which does not meet the requirement of real time, low latency, and high quality of service (QoS) in the network of thousands of IoT devices and affects the overall efficiency of the system [6, 7].

Due to the fact that traditional cloud computing poses many challenges, mobile edge computing (MEC) is proposed, which consists of relatively weak edge devices [8–10]. MEC is a novel paradigm that extends cloud computing capabilities and services to the edge of the network. On one hand, MEC ensures that data processing mainly depends on the local devices rather than the cloud servers. On the other hand, MEC usually does not need to establish a relationship with remote cloud servers; it can meet most requirements of local users very well [11]. However, MEC does not have enough computing capability compared to cloud servers, since it only includes limited computing devices. Once computing capability of a single edge device is exceeded, MEC requires other edge devices to assist or forward residual data to the cloud servers for processing, so that the system can still maintain good performance with the rapid growth of the number of users or the amount of data.

In this paper, firstly, we propose a novel three-layer network architecture model which combines cloud computing and edge computing (abbreviated as CENAM), and model the communication delay and computation delay of each part of the CENAM. Then, in edge computing, we propose a computational scheme of mutual cooperation between the edge devices based on the weighted undirected graph and use the Kruskal algorithm and Lagrange multiplier method to solve the communication delay and computation delay, respectively. In addition, we focus on the balanced transmission method to solve the data transmission delay from edge devices to cloud servers and solve the computation

delay of cloud servers according to their characters. Finally, based on simulations and numerical results, we show the better performance of CENAM in terms of reducing data processing delay.

The rest of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we describe the composition of CENAM. Section 4 explains in detail the computational scheme of CENAM that we proposed. Section 5 is algorithm solving. We analyze the performance of our solution in Section 6. In Section 7, we conclude the paper and present some future work.

2. Related Work

The related research of MEC has received considerable attention in recent times. For example, Subramanya et al. [12] proposed a resource constrained cloud-enabled small cell that includes a MEC server for deploying mobile edge computing functionalities and presented the architecture with special focus on realizing the proper forwarding of data packets between the mobile data path and the MEC applications. Sharma and Wang [13] put forward a novel framework for coordinated processing between edge and cloud computing by integrating advantages from both the platforms, which can provide real-time information to end users and enhance the performance of wireless IoT networks. Masip-Bruin et al. [14] introduced a layered F2C architecture with benefits and strengths and provided the real need for their coordinated management. To adapt to rapid increment of mobile resources, Lee et al. [15] proposed collaborate platform solution, which greatly simplified the development mechanism of mobile collaborative applications and reduced the delay and energy consumption. Due to the problem that the mobile users are far away from the cloud servers and generate great transmission delay, Intharawijitr et al. [16] relaxed the delay constraint system and selected the edge servers to reduce the system delay effectively. Hu et al. [17] focused on the services allocation problem in MEC and found trade-offs between average network delay and load balance. Simulations showed that the variance of the load on MEC servers is reduced by 18.9% with nearly the same network delay. In order to achieve intelligent D2D communication, Bello and Zeadally [18] focused on intelligent routing protocols and presented an overview of how intelligent D2D communication can be achieved in the IoT ecosystem.

Fog computing [19–21], one of the typical representations of edge computing, has attracted more and more attention to solve related problems. Sarkar and Misra [22] established a theoretical model of fog computing' architecture and defined the structural components mathematically. Then, they compared it to the traditional architecture of cloud computing and analyzed the performance of delay and energy consumption in the background of IoT. Deng et al. [23] focused on the problem of delay and optimal energy consumption in cloud-fog computing. They divided the problem of data transmission, existing in fog computing, into three subproblems and analyzed the performance of delay from the perspective of balanced load. But none of them considered the collaboration between edge devices. According to the problem of high processing delay when cloud computing is used in the medical data scene, He and Ren [24] proposed cloud-fog network architecture model for medical big data. Facing the problem of weak computing capability of edge devices, they put forward a distributed computational scheme. Pham and Huh [25] formulated the task scheduling problem in cloud-fog environment and proposed a heuristic-based algorithm. As a result, they achieved the balance between the makespan and the monetary cost of cloud resources.

3. The Description of CENAM

In the real Internet of Things applications, users want to get the information that they want and transmit it as soon as possible, so the information providers need to have high working efficiency. Facing high transmission delay and computing pressure of cloud data center, we propose CENAM, as shown in Figure 1.

Network architecture model is mainly divided into three layers: cloud computing layer, edge computing layer, and mobile terminal layer, where the bottom is mobile terminal layer, which includes all mobile terminal devices, such as smartphones, laptops, and cars. Users accessing the network through intelligent devices not only can quickly access services from the network architecture, but also can store their own information in the data center of network architecture.

The middle layer is edge computing layer, which consists of edge devices (routers, gateways, switches, and access points). Edge devices are mainly distributed in local mobile subscriber premises, such as parks, shopping centers, and buses. Edge computing layer is the bridge between cloud servers and end users. Besides computation and storage of the local data, the residual data that the edge computing layer cannot handle is forwarded to the cloud computing layer for processing.

The top layer is cloud computing layer, which consists of high-end servers and data center. It has powerful computing and storage capabilities and is responsible for computing and storing residual data that the edge computing layer cannot handle.

4. Computational Scheme of CENAM

As shown in Figure 1, because the edge devices are close to end users and accept service requests from them through local area network (LAN), the LAN communication delay can be ignored (compared to WAN). Unprocessed data in MEC is forwarded to cloud servers via the WAN for processing, so communication delay from edge devices to cloud servers needs to be considered. In the process of data processing, we mainly consider the delay of edge computing layer and cloud computing layer including communication delay from edge devices to cloud servers.

4.1. Delay Computational Scheme in Edge Computing Layer. Edge devices are mainly distributed in areas such as supermarkets, parks, and buses. The cooperation between them is inseparable. In this paper, we abstract a network topology

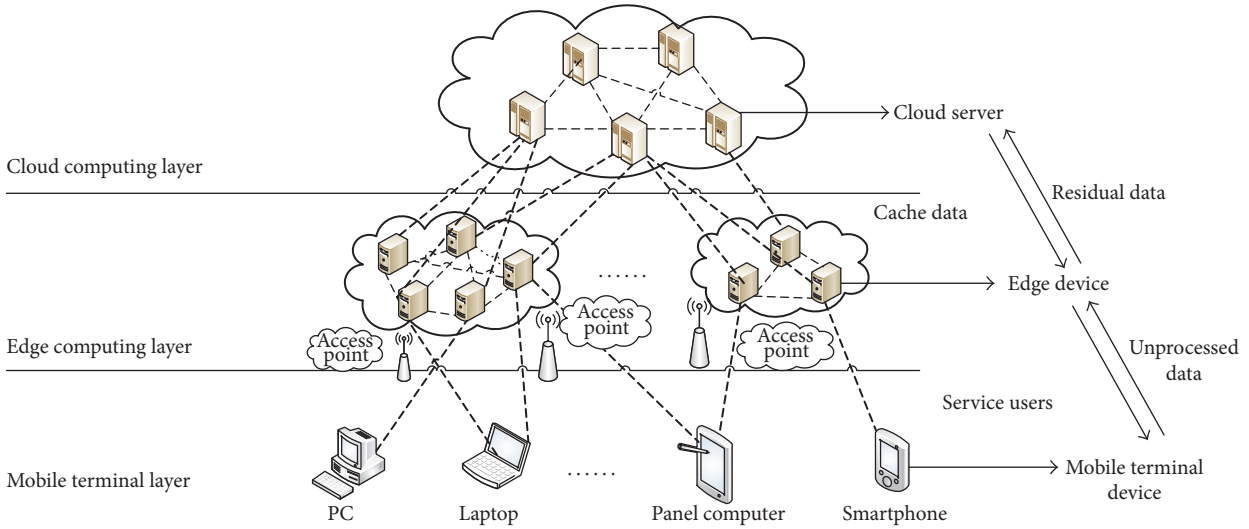

Figure 1: CENAM.

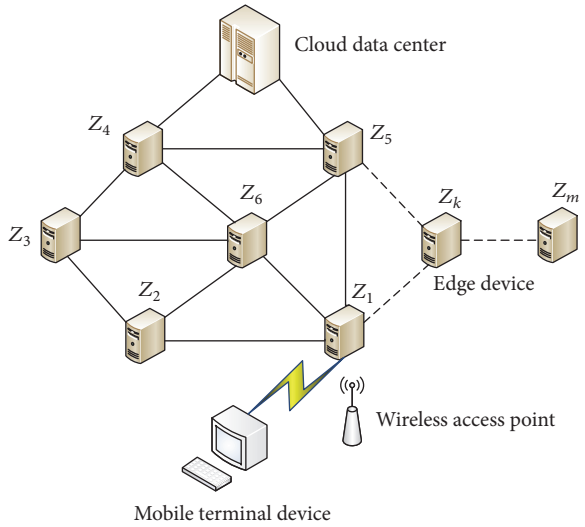

Figure 2: Network topology graph of edge devices.

graph consisting of m edge devices (shown in Figure 2) as a weighted undirected graph (shown in Figure 3).

$$V = \{z_1, z_2, \ldots, z_i, \ldots, z_m\} \tag{1}$$

is a vertex set, where z_i is a set of edge devices and m is the number of edge devices.

$$E = \left\{e_{z_1,z_2}, \ldots, e_{z_i,z_j}, \ldots, e_{z_{m-1},z_m}\right\} \tag{2}$$

is an edge set, where e_{z_i,z_j} is the communication link between edge nodes z_i and z_j. τ_{z_i,z_j} on the edge is the communication delay between edge nodes z_i and z_j, and the size of its value is randomly generated with reference to the numerical value of [24].

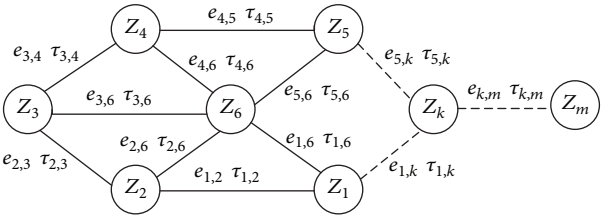

Figure 3: Weighted undirected graph of edge devices.

We suppose that the computing capability of each edge node z_i in Figure 3 is

$$v_{z_i} \quad (i = 1, 2, \ldots, m) \tag{3}$$

and the computation task of edge computing layer is X. In order to reduce the amount of data forwarded from edge computing layer to cloud computing layer to reduce communication delay, we need to enhance the computing capability of edge node and process data on edge network as much as possible. Therefore, we propose a scheme of cooperation between edge devices. In the process of data processing, edge device z_i receives the computation task X from end users and divides the task into several subtasks that meet $x_i = \delta_i X$ and then assigns the subtasks to edge nodes which included themselves to process; the specific scheme is as follows.

The delay of edge computing layer includes the communication delay between edge nodes and the computation delay of edge nodes. For the communication delay, in weighted undirected graph consisting of edge nodes, we regard the communication delay between edge nodes as the weight and use the Kruskal algorithm to generate a minimal spanning tree, so we get the minimum weight $W(T)$, that is, the minimum communication delay. For edge device i, its computation delay can be described by the amount of computation x_i assigned to it, which satisfies two points: (i)

As the amount of computation increases, the computation delay of edge devices increases accordingly. (ii) The more the amount of computation increases, the faster the computation delay of edge devices increases. Therefore, the following function is used to describe the computation delay of edge device i.

$$D_i^{\text{Edge}} = \frac{1}{v_{z_i}} a_i x_i^2, \tag{4}$$

where v_{z_i} is the computing capability of edge device i, x_i is the amount of data to be processed by edge device i, and a_i is a predetermined real number between 0 and 1.

So, the computation delay of m edge nodes can be expressed as

$$\begin{aligned} D_{\text{Edge}}^{\text{com}} = \min \quad & \sum_{i=1}^{m} \frac{1}{V_{Z_i}} a_i x_i^2 \\ \text{s.t} \quad & \sum_{i=1}^{m} x_i = X \quad 0 < x_i < x_i^{\max}, \end{aligned} \tag{5}$$

where $x_i^{\max}$ is the maximum amount of data that edge devices i can handle and X is the total amount of data that the edge computing layer needs to process. So the amount of data processed on edge computing nodes can form an m-dimension vector

$$X = [x_1, x_2, \ldots, x_m]. \tag{6}$$

Thus, the following objective function is established for the delay generated in edge computing layer:

$$\begin{aligned} D_{\text{Edge}} = \min \quad & \left(\sum_{i=1}^{m} \frac{1}{V_{z_i}} a_i x_i^2 \right) + W(T) \\ \text{s.t} \quad & \sum_{i=1}^{m} x_i = X \quad 0 < x_i < x_i^{\max}. \end{aligned} \tag{7}$$

4.2. Delay Computational Scheme in Cloud Computing Layer. Delay in cloud computing layer includes the computation delay of cloud servers and the communication delay from edge devices to cloud servers. In cloud computing layer, we assume that there are n cloud servers; the amount of data processed on cloud servers is Y. For the cloud server j, if the amount of data it needs to deal with is y_j and the computing capability is v_j, its computation delay can be expressed as

$$D_j^{\text{com}} = \frac{y_j}{v_j} \quad (j = 1, 2, \ldots, n), \tag{8}$$

So, when the n cloud servers deal with the amount of data being Y, the computation delay in cloud computing layer is

$$\begin{aligned} D_{\text{cloud}}^{\text{com}} = & \sum_{j=1}^{n} \frac{y_j}{v_j} \quad (j = 1, 2, \ldots, n) \\ \text{s.t} \quad & \sum_{j=1}^{n} y_j = Y \quad y_j \geq 0, \end{aligned} \tag{9}$$

If the data loss rate is not considered, the delay of the WAN transmission path from the edge device i to the cloud server j is d_{ij}, and the traffic rate is λ_{ij}. According to [23], the communication delay is

$$D_{ij}^{\text{comm}} = d_{ij} \lambda_{ij} \quad (i = 1, 2, \ldots, m \;\; j = 1, 2, \ldots, n). \tag{10}$$

Therefore, the delay of transmitting data from edge devices to cloud servers in CENAM can be expressed as

$$\begin{aligned} D_{\text{Edge-Cloud}}^{\text{comm}} = & \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \lambda_{ij} \\ & (i = 1, 2, \ldots, m \;\; j = 1, 2, \ldots, n) \\ \text{s.t} \quad & \sum_{i \in M} \lambda_{ij} = y_j \\ & \sum_{j \in n} \lambda_{ij} = x_i \\ & 0 \leq \lambda_{ij} \leq \lambda_{ij}^{\max}, \end{aligned} \tag{11}$$

where $\lambda_{ij}^{\max}$ is the maximum traffic rate limited by bandwidth.

4.3. Problem Conception. In the CENAM, system delay mainly includes the computation delay of edge devices and cloud servers and the communication delay from edge devices to cloud servers, so the system delay is defined as

$$D^{\text{sys}} = D_{\text{Edge}} + D_{\text{Cloud}}^{\text{com}} + D_{\text{Edge-Cloud}}^{\text{comm}}. \tag{12}$$

Considering the delay optimization of CENAM and the system load balancing, the objective function is established as

$$\begin{aligned} \min \quad & D^{\text{sys}} \\ \text{s.t} \quad & L = X + Y \\ & \sum_{i=1}^{m} x_i = X \\ & \sum_{j=1}^{n} y_j = Y, \end{aligned} \tag{13}$$

where L is the total amount of data processed in system, X is the amount of data processed in edge computing layer, and Y is the amount of data processed in cloud computing layer, and they meet $L = X + Y$.

5. Algorithm Solution

5.1. Solution of Delay Optimization in Edge Computing Layer

5.1.1. Solution of Communication Delay in Edge Computing Layer. We use Kruskal algorithm to solve the problem of minimum communication delay between edge node. For the weighted undirected graph $G = (V, E)$, its minimum spanning tree is

$$T = (U, TE), \tag{14}$$

and the initial state is

$$U = V,$$
$$TE = \{\}. \tag{15}$$

In this way, each edge node in T constitutes a connected component. Then, according to the order of the weight of edges from small to big, the edges of set E are investigated in turn. If the two vertices under investigation are two different connected components in T, the investigated edges are added to TE, and the two connected components are joined to one. If not, the edge is removed to avoid the loop. So moving on, when the connected component in T is one, it is the minimal spanning tree of G. Kruskal algorithm is described as follows:

(1) Initialization: $U = V$; $TE = \{\}$.

(2) Repeat the following operations until the number of connected components in T is one.

Find the shortest edge (u, v) in E.

If vertices u and v are located in two different connected components in T, then

(i) incorporate edge (u, v) into TE,

(ii) merge two connected components into one.

In T, we mark edge (u, v) and make it not participate in the selection of the subsequent minimal edges.

(3) Finally, we get the minimum communication delay $W(T)$.

5.1.2. Solution of Computation Delay in Edge Computing Layer. The target function of computation delay is

$$f(x) = \min \sum_{i=1}^{m} \frac{1}{V_{z_i}} a_i x_i^2,$$
$$h_i(x) = \begin{cases} 0 < x_i < x_i^{\max} \\ \sum_{i=1}^{m} x_i = X. \end{cases} \tag{16}$$

We solve this equality constrained optimization problem based on the Lagrange multiplier method, and the specific steps are as follows.

(1) Given the initial point $x^{(0)}$, the initial multiplier vector v $(v_1, v_2, \ldots, v_m)$, the penalty factor $M > 0$, the amplification factor $a > 0$, and the accuracy $\varepsilon > 0$, the parameter $\gamma \in (0, 1)$ and $k = 1$.

(2) Construct an objective function as follows:

$$F(x) = f(x) + \frac{M}{2} \sum_{i=1}^{m} h_i(x)^2 - \sum_{i=1}^{m} v_i h_i(x). \tag{17}$$

$f(x)$ is target function and $h_i(x)$ is constraint function.

(3) Use the unconstrained nonlinear programming method (Newton method is used in this paper), $x_{(k-1)}$ is used as the initial point to solve min $F(x)$, and the optimal solution is $x_{(k)}$.

(4) If

$$\left\| h(x_{(k)}) \right\| < \varepsilon, \tag{18}$$

then stop iterating and output $x_{(k)}$; otherwise go to step (5).

(5) If

$$\frac{\left\| h(x_{(k)}) \right\|}{\left\| h(x_{(k-1)}) \right\|} \geq \gamma, \tag{19}$$

then $M = \alpha M$; otherwise M remains unchanged; go to step (6).

(6) Define

$$v_i = v_i - M h_i(x_{(k)}), \quad i = 1, 2, \ldots, m, \tag{20}$$

set $k = k + 1$, and go to step (2).

So, we obtain the optimal solution $x_{(k)}$ and the minimum computation delay of edge computing layer. And the optimal delay of the edge computing layer is the sum of the computation delay and communication delay.

5.2. Solution of Delay Optimization in Cloud Computing Layer. According to the above analysis, the delay of cloud computing layer mainly includes the computation delay of cloud servers and the communication delay from edge devices to cloud servers. Based on the load balancing principle, the communication delay problem is regarded as a balanced transmission problem. We suppose that the amount of data that is not processed in edge computing layer (i.e., the amount of data that needs to be processed in cloud computing layer) is l_i and the amount of data that each cloud server can deal with is y_j. The amount of data that is not processed in edge computing layer needs to be forwarded to cloud computing layer for processing. In order to minimize the transmission delay, an optimal transmission scheme is needed. The delay of the WAN transmission path from the edge device i to the cloud server j is d_{ij}, and the following delay matrix can be obtained:

$$d_{ij} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdots & \cdots & & \cdots \\ d_{m1} & d_{m2} & & d_{mn} \end{bmatrix}. \tag{21}$$

λ_{ij} is the traffic rate from edge device i to cloud server j, and the following transmission matrix can be obtained:

$$\lambda_{ij} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2n} \\ \cdots & \cdots & & \cdots \\ \lambda_{m1} & \lambda_{m2} & & \lambda_{mn} \end{bmatrix}. \tag{22}$$

Differing from [23], we use the method of balanced transmission to solve the problem of communication delay, and obtain the optimal allocation matrix Z_{ij}. Z_{ij} is the

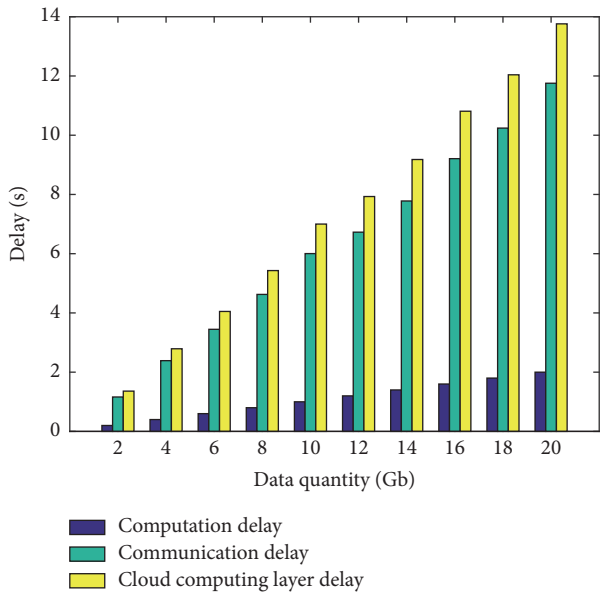

Figure 4: Computation and communication delay of cloud computing layer.

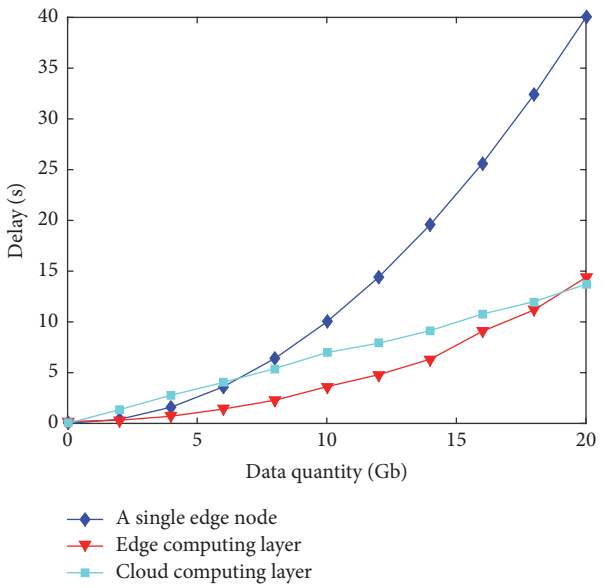

Figure 5: Comparison of data processing delays.

amount of transmission data from edge device i to cloud server j.

$$Z_{ij} = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \cdots & \cdots & & \cdots \\ Z_{m1} & Z_{m2} & \cdots & Z_{mn} \end{bmatrix}. \quad (23)$$

For the computation delay of cloud servers, we do not consider the data loss rate. The principle of conservation of $\sum_{i \in M} \lambda_{ij} = y_j$ shows that the amount of data that needs to be processed by cloud server is y_j. In experiment, we assume that the number of cloud servers n is 5 and the computing capability of cloud server is 10 GHz. We adopt MATLAB experimental platform, and the data in experiment is all simulated. We obtain the communication delay and computation delay by MATLAB simulation, as shown in Figure 4.

Figure 4 shows that, in cloud computing layer, the computation delay is relatively small and varies little because of the strong computing capability of cloud server. The delay of cloud computing layer is mainly caused by the communication delay from edge devices to cloud servers. On one hand, this is because the communication with long distance from end users to the cloud data center may generate high delay. On the other hand, the limitation of network bandwidth increases the transmission delay from edge devices to cloud servers greatly. As the amount of data increases continuously, the communication delay increases faster.

6. Simulation Results and Analysis

In order to verify the effectiveness of the scheme of data processing delay in edge computing layer, we compare the data processing delay performance to a single edge node and the cloud computing layer at first. Then, we analyzed the impact of the proportion of the amount of data a processed in edge computing layer on data processing delay. Finally, the influence of the number of edge nodes on data processing delay is analyzed.

The experimental platform adopts MATLAB, and the computing capability and communication delay of edge devices and cloud servers are all found in [24]. The data in experiment is set by simulation, the number of edge nodes m is 10, and the number of cloud servers n is 5. Besides, we set the computing capability of cloud servers to 10 GHz. The computing capability of each edge device is shown in Table 1.

6.1. Performance Analysis of Data Processing Delays in Edge Computing Layer. In edge computing layer, we propose the scheme of computation delay and communication delay. In order to verify its effectiveness in data processing delay, we compare the delay to a single edge node and cloud computing layer. The experimental results are shown in Figure 5.

Experimental results show that when the amount of data is $X < 2$ Gb, a single edge node has less delay compared to the cloud computing layer and the edge computing layer; this is because it does not produce communication delay, and the amount of data is within the range of computing capability of a single edge node. However, as the amount of data increases, the delay caused by the computing capability of the single edge node will increase rapidly. Cloud servers have strong computing capability, but the end users are far away from them and limited by bandwidth, which will generate great communication delay, so the delay of data processing is higher than that of edge computing layer. At this time, the scheme of cooperation between multiple edge nodes in edge computing layer shows better performance. When the amount of data is $X > 19$ Gb, the delay in edge computing layer will be affected by the limitation of computing capability of

Table 1: The computing capability of edge nodes.

Parameter	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}
v_i/GHz	2	1	1	0.5	0.4	0.7	0.6	0.3	0.5	1

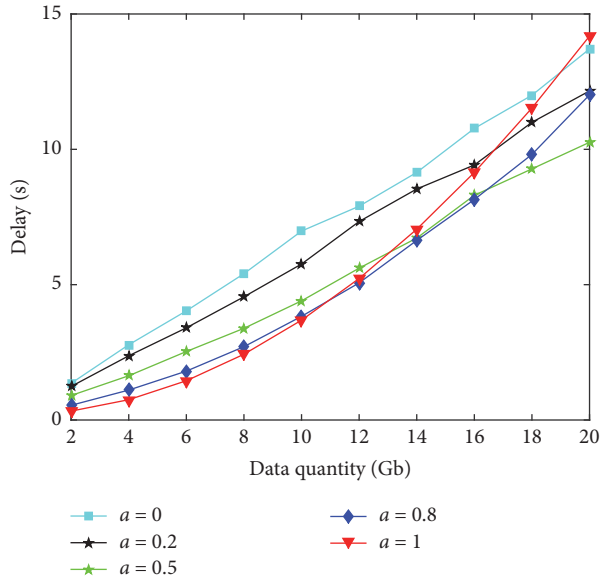

Figure 6: The impact of a on data processing delay.

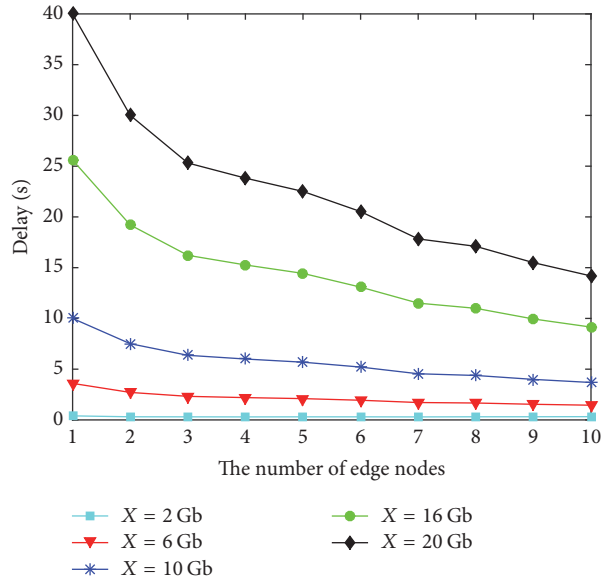

Figure 7: The influence of the number of edge nodes on data processing delay.

the single edge node, and there is a significant rise in data processing delay. However, cloud computing layer with its powerful computing capability makes data processing delay less than that of edge computing layer. Therefore, it is possible to put the appropriate amount of data to edge computing layer to reduce the delay.

6.2. The Impact of a on Data Processing Delay. a is the percentage of data that is processed in edge computing layer. To verify the performance of CENAM, we study the impact of a on data processing delay. Simulation is shown in Figure 6.

The experimental results show that when $a \leq 0.5$, that is, the amount of data processed in edge computing layer is less than half, the larger the a, the smaller the delay. When $a > 0.5$ and the amount of data is small, the larger the a, the smaller the delay. However, as the amount of data increases, the delay will increase accordingly. And the larger the a, the faster the corresponding delay, and it even exceeds the traditional cloud computing layer delay. This is because when the amount of data reaches a threshold (e.g., when $a = 0.8$, the threshold is 16 Gb; when $a = 1$, the threshold is 13 Gb), data processing delay increases rapidly and exceeds a certain value because of the limitation of computing capability of the single edge node. This shows that data processing delay in edge computing layer is limited by computing capability of the single edge node, and when the amount of data increases to a certain degree, the delay will increase. It also shows that when the total amount of data is small, we can process them in edge computing layer and generate small delay. However, when there is a large amount of data, the cooperation between edge devices and cloud servers can effectively reduce the data processing delay.

6.3. The Influence of the Number of Edge Nodes on Data Processing Delay. In order to study the influence of the number of edge nodes on data processing delay in edge computing layer, we solve the data processing delay when the total amount of data is 2 Gb, 6 Gb, 10 Gb, 16 Gb, and 20 Gb. The result is shown in Figure 7.

The experimental results show that, with the increase of the number of edge nodes, the overall data processing delay shows a downward trend. When the amount of data is small (as shown below 6 Gb), the increase in the number of edge nodes has less impact on data processing delay, and it is basically stationary. When the amount of data is large (such as from 10 Gb to 20 Gb), with the increase of the number of edge nodes, there is a significant decrease in data processing delay. That is because when the amount of data is small, the computation delay of edge nodes is small, and the communication delay will become large with the increase of edge nodes. So the data processing delay mainly depends on the communication delay between edge nodes, but the change is not obvious. When the amount of data is large, the computation delay of edge nodes increases accordingly, and the communication delay between edge nodes is relatively stable when the bandwidth is allowed. The delay of data processing mainly depends on the computation delay of edge nodes. Therefore, as the number of edge nodes increases, the amount of data processed by each edge node will decrease

and the delay will also be reduced, so that the overall data processing delay will decrease significantly. This shows that, according to the computing capability of edge nodes, it is important to determine the appropriate number of edge nodes to reduce the delay of data processing.

7. Conclusion and Future Work

In this paper, the concept of CENAM is proposed to solve the problem of high delay of data processing in traditional cloud computing. In edge computing layer, we use the method of the cooperation between multiple edge nodes to improve data processing capability and reduce the computation delay of edge computing layer. Besides, we use the Kruskal algorithm to solve the communication delay between edge nodes. In cloud computing, the communication delay from edge devices to cloud servers is reduced based on the way of balanced transmission. Simulation shows that the CENAM proposed in this paper can effectively reduce the data processing delay and perform better than both the single edge node and traditional cloud computing. In the future work, we will continue to study the influence of the location allocation and service mode of edge devices on data processing delay and energy consumption, so that we can further improve the performance of system.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61672321 and 61771289), the Shandong Provincial Graduate Education Innovation Program (SDYY14052 and SDYY15049), the Shandong Provincial Specialized Degree Postgraduate Teaching Case Library Construction Program, the Shandong Provincial Postgraduate Education Quality Curriculum Construction Program, the Shandong Provincial University Science and Technology Plan Projects (J16LN15), and the Qufu Normal University Science and Technology Plan Projects (xkj201525).

References

[1] W. Fang, "Paradigm shift from cloud computing to fog computing," *Journal of Nanjing University of Information Science Technology*, vol. 8, no. 5, pp. 404–414, 2016.

[2] A. J. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: Progress and challenges," in *Proceedings of the 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2016*, pp. 83-84, UK, April 2016.

[5] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to Networking Cloud and Edge Datacenters in the Internet of Things," *IEEE Cloud Computing*, vol. 3, no. 3, pp. 64–71, 2016.

[6] S. C. Li, "Research on Internet of Things Architecture Based on Fog Computing," *Telecommunication Engineering Technology and Standardization*, vol. 30, no. 11, pp. 63–66, 2017.

[7] P. Corcoran and S. K. Datta, "Mobile-Edge Computing and the Internet of Things for Consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73-74, 2016.

[8] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75–87, 2017.

[9] M. Satyanarayanan, "The emergence of edge computing," *The Computer Journal*, vol. 50, no. 1, Article ID 7807196, pp. 30–39, 2017.

[10] J. Tang and T. Q. S. Quek, "The role of cloud computing in content-centric mobile networking," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 52–59, 2016.

[11] S. K. Datta and C. Bonnet, "An edge computing architecture integrating virtual IoT devices," in *Proceedings of the 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*, pp. 1–3, NAGOYA, Japan, October 2017.

[12] T. Subramanya, L. Goratti, S. N. Khan, E. Kafetzakis, I. Giannoulakis, and R. Riggio, "A practical architecture for mobile edge computing," in *Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software-Defined Networks (NFV-SDN)*, pp. 1–4, Berlin, November 2017.

[13] S. K. Sharma and X. Wang, "Live Data Analytics with Collaborative Edge and Cloud Processing in Wireless IoT Networks," *IEEE Access*, vol. 5, pp. 4621–4635, 2017.

[14] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications Magazine*, vol. 23, no. 5, pp. 120–128, 2016.

[15] J. Lee, H. Lee, Y. C. Lee, H. Han, and S. Kang, "Platform Support for Mobile Edge Computing," in *Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 624–631, Honolulu, CA, USA, June 2017.

[16] K. Intharawijitr, K. Iida, H. Koga, and K. Yamaoka, "Practical Enhancement and Evaluation of a Low-Latency Network Model Using Mobile Edge Computing," in *Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 567–574, Turin, July 2017.

[17] B. Hu, J. Chen, and F. Li, "A dynamic service allocation algorithm in mobile edge computing," in *Proceedings of the 2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 104–109, Jeju Island, Korea (South), October 2017.

[18] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the internet of things," *IEEE Systems Journal*, 2014.

[19] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, "Fog Computing: Focusing on Mobile Users at the Edge," *Computer Science*, pp. 1–7, 2015.

[20] I. Stojmenovic and S. Wen, "The fog computing paradigm: scenarios and security issues," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '14)*, pp. 1–8, IEEE, Warsaw, Poland, September 2014.

[21] Z. H. Yang, "Boundary computation model of Internet of things: fog computing," *Internet of things technology*, vol. No. 12, pp. 65–67, 2014.

[22] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, 2016.

[23] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[24] L. X. He and Z. Y. Ren, "Cloud network for medical large data and its distributed computing scheme," *Journal of Xi'an Jiaotong University*, vol. 50, no. 10, pp. 71–77, 2016.

[25] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium, APNOMS 2016*, Japan, October 2016.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7308913, 9 pages
https://doi.org/10.1155/2018/7308913

Research Article

Method of Resource Estimation Based on QoS in Edge Computing

Guangshun Li, Jianrong Song, Junhua Wu, and Jiping Wang

School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

Correspondence should be addressed to Guangshun Li; 30752585@qq.com

Received 13 October 2017; Accepted 21 December 2017; Published 22 January 2018

Academic Editor: Shangguang Wang

With the development of Internet of Things, the number of network devices is increasing, and the cloud data center load increases; some delay-sensitive services cannot be responded to timely, which results in a decreased quality of service (QoS). In this paper, we propose a method of resource estimation based on QoS in edge computing to solve this problem. Firstly, the resources are classified and matched according to the weighted Euclidean distance similarity. The penalty factor and Grey incidence matrix are introduced to correct the similarity matching function. Then, we use regression-Markov chain prediction method to analyze the change of the load state of the candidate resources and select the suitable resource. Finally, we analyze the precision and recall of the matching method through simulation experiment, validate the effectiveness of the matching method, and prove that regression-Markov chain prediction method can improve the prediction accuracy.

1. Introduction

With the development of Internet of Things (IoT) [1], more and more devices, especially mobile devices, constantly access the Internet. CISCO predicts that 50 billion devices will connect to the Internet by 2020 [2]. These devices will generate large amounts of data at the end of the network, which leads to the increment of the burden of cloud data center. Moreover, the remote distance between the mobile devices and cloud data center makes the transmission delay increase, which makes some delay-sensitive services can not get response and processing rapidly. IoT services, such as connected vehicle and video streaming, require high bandwidth and low latency content delivery to guarantee QoS. Edge computing [3] plays an important role by using network resources near the local network to provide a low latency service and improve QoS.

Edge computing is located between the end devices and cloud data center. It transfers data and computing power from the cloud to the edge of the network, which processes data locally [4]. Due to the fact that edge servers are close to end users, the latency between mobile devices and data center is reduced. Moreover, the resources are close to the user that can improve QoS to some extent.

In the IoT environment, the number of network devices is increasing constantly. The resources with the same service function are also increasing gradually. The complexity of user requirements makes resource management a challenge. Among the many available resources, selecting the suitable resources to meet the needs of users has become the main goal.

Although the latency is reduced, the resources in edge computing are limited compared to cloud computing. The availability and short-term prediction of resources load [5, 6] have some influence on task scheduling, application performance, and the QoS of users. The method of resource prediction can provide the appropriate resource for users by analyzing the load of the resource itself. Therefore, the estimation of Resource QoS is significant to user satisfaction and task allocation in edge computing.

In this paper, we first propose an edge computing framework. Then, we introduce the penalty factor and Grey incidence matrix to improve the accuracy of similarity matching and select resources which satisfy the needs of users. We use the regression-Markov chain prediction method for candidate resources to analyze the change of the load state of

the resource itself. Finally, we prove the effectiveness of the estimation method through simulation experiment.

The remainder of this paper is organized as follows. The related works are introduced in Section 2. We describe the architecture of the edge computing in Section 3. In Section 4, we describe the method of dynamic resource estimation. The experiment results are presented in Section 5. Section 6 concludes the paper.

2. Related Work

Edge computing is the extension of cloud computing to network edge. There is no standard architecture. The work of resource estimation has not been well addressed. Mao et al. [7] developed a joint radio and computational resource management algorithm for multiuser MEC systems, with the objective of minimizing the average weighted sum power consumption of the mobile devices and the MEC server. Shekhar and Gokhale [8] proposed Dynamic Data Driven Cloud and Edge Systems (D3CES) as a framework for adaptive resource management across cloud and edge resources to provide QoS guarantees for performance-sensitive applications. Wang et al. [9] developed an Edge NOde Resource Management (ENORM) framework to manage edge nodes. They proposed a mechanism for provisioning and autoscaling edge node resources. Their method reduced data transmission and communication frequency between the edge node and cloud.

Aazam and Huh [10] proposed a dynamic resource estimation and pricing model for IoT. Their work was mainly focused on considering different types of customers and devices, and the service-oriented relinquish probabilities. Based on previous studies, Aazam et al. [11] proposed a QoE-based resource estimation model to enhance QoS. They devised MeFoRE method on the basis of service relinquish probability and historical records. And they considered previous QoE and Net Promoter Score records to enhance QoS. Zuo et al. [12] proposed a resource estimation model based on entropy optimization and dynamic weighted method to accurately grasp the dynamic load and availability information of resources. They used the objective function and constraint condition of maximum entropy to select the resources which meet the QoS of user. This method achieved optimal scheduling and guaranteed the QoS of user. Dynamic weighted load evaluation was carried out on the filtered resources to realize load balancing and improve system utilization.

Zhao et al. [13] proposed a multidimensional resource model for dynamic resource matching in the IoT, with the multidimensional description of the IoT resources. They considered the ontology structure and ontology description to calculate the similarity by matching the hardware features and software feature between resources. Zhou et al. [14] proposed a multiple QoS resource selection and estimation algorithm. They used Multiple Attribute Decision Making and Analytic Hierarchy Process (AHP) algorithm. Their method mainly considered the preference information of users. Dong et al. [15] analyzed the QoS of a single resource node. They used a local optimization algorithm to select the appropriate cloud resources that meet the needs of users. Ding et al. [16] proposed QoS-aware resource matching and recommendation method in cloud computing. They described a resource matching algorithm that considers both functional requirements and QoS attributes and designed a resource recommendation method for cloud computing.

3. Edge Computing

Edge computing, located between the mobile end devices and cloud data center, provides computing, storage, and network services. Edge computing has the ability of decentralized computation and storage compared with cloud computing. The resources near the user can support mobile devices for real-time communication. The main goal of edge computing is to preprocess data, reduce the delay, and serve the applications of low delay and real-time response. Currently, the common edge computing architecture is a three-tier network architecture, as shown in Figure 1.

As the underlying node, the end devices not only consume data but also produce data. In particular, mobile devices will require more resource from edge servers rather than cloud data center. The end devices include all IoT devices, such as smart mobile phone, intelligent vehicle, and virtual sensor nodes. These devices are able to sense data, communicate with the upper layer through sensor networks, 3G, WiFi, and so on, and transmit the collected raw data. Compared with the servers in data centers, most of the IoT devices or sensors at the edge are somewhat limited in both computing power and battery capacity [17]. Edge devices include some traditional network devices (routers, switches, etc.) and some specially deployed devices (local servers). Edge devices have the ability of computing, processing, storage, and forwarding data to the cloud servers. Edge servers can be connected directly to nearby mobile devices via one-hop wireless link. Micro data center (MDC) resources include computing resources, network resources, storage resources, and software resources. On the one hand, resources come from the cached local resources; on the other hand, they come from the data obtained by the monitoring equipment. As the top layer, cloud computing layer includes data center (DC) and some cluster servers. The cloud servers can receive data sent by edge devices for processing and storage.

Figure 2 shows the request process of the service in the edge computing. The user first submits the request to the system administrator through the end devices. The system administrator stores the submitted information and transmits the statistical QoS requirements to the edge servers through the edge gateway. Monitoring equipment generates statistics log by tracking resource utilization and availability of sensor, applications, and services. The monitoring data is transmitted to the edge servers. The edge servers analyze the QoS of requested service for end users and process locally. The service is divided into several tasks which are processed locally as far as possible. Each task selects the best matching resource from the resource pool based on the estimation results. Then, the selected resource is allocated and scheduled to meet the QoS requirements.

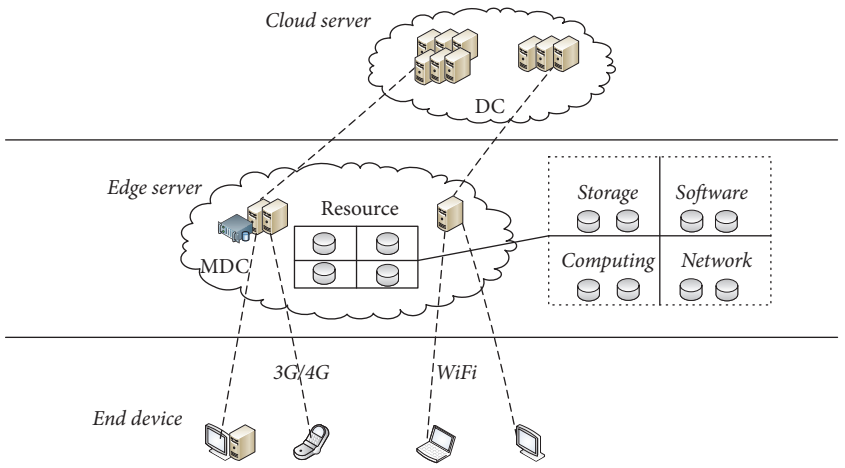

Figure 1: Edge computing architecture.

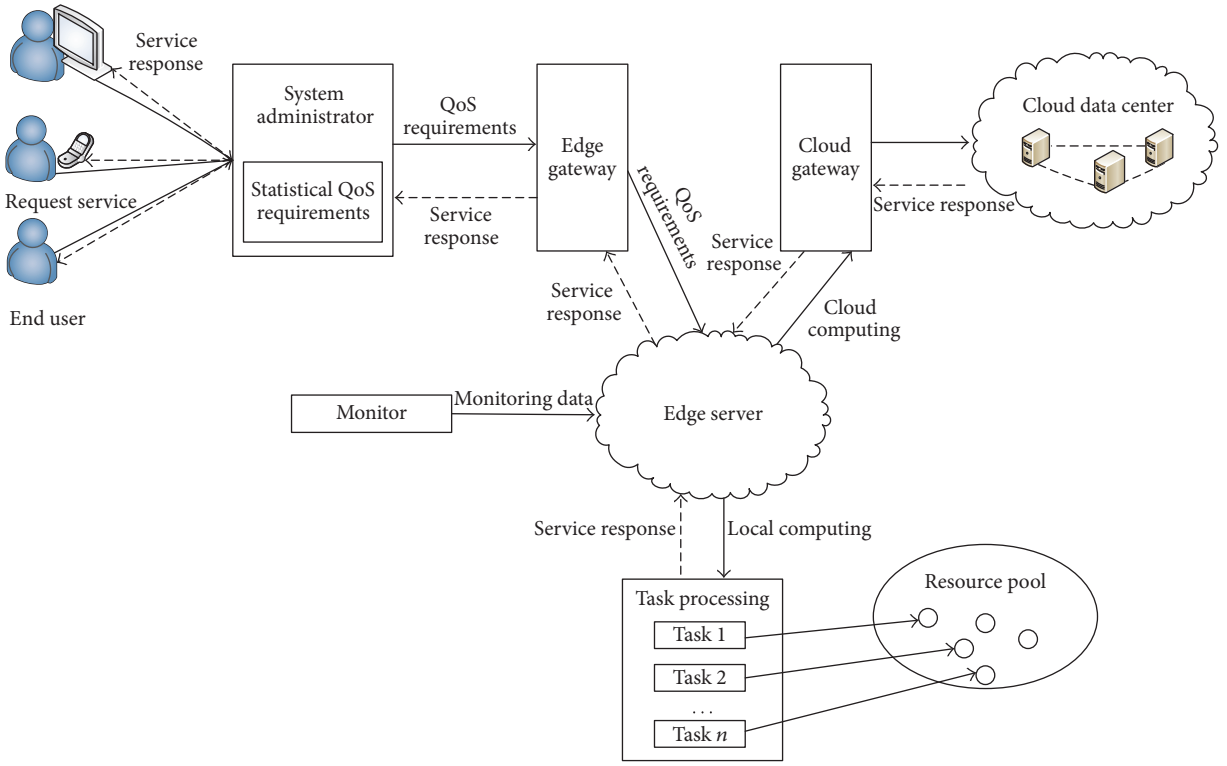

Figure 2: The request process of service in edge computing.

4. Dynamic Estimation Method

In this section, we describe the dynamic estimation method and estimate the resource with the same service function. The specific process is shown in Figure 3.

As shown in the figure, the dynamic estimation method includes two phases: similarity matching algorithm and regression-Markov chain prediction method. First, we establish a Resource QoS matrix to calculate the matching degree between QoS requirements of users and resources through similarity matching. Then, resources which satisfy the needs of the users are selected by threshold. Finally, we analyze the change of resource load and select the optimal resource through regression-Markov chain prediction method.

4.1. Similarity Matching Method

4.1.1. Resource Description. The resources in the edge nodes can be provided to meet the QoS requirements of users. In this paper, resource set which have the same service function is defined as follows:

$$Rs = \{r_1, r_2, \ldots, r_n\}. \quad (1)$$

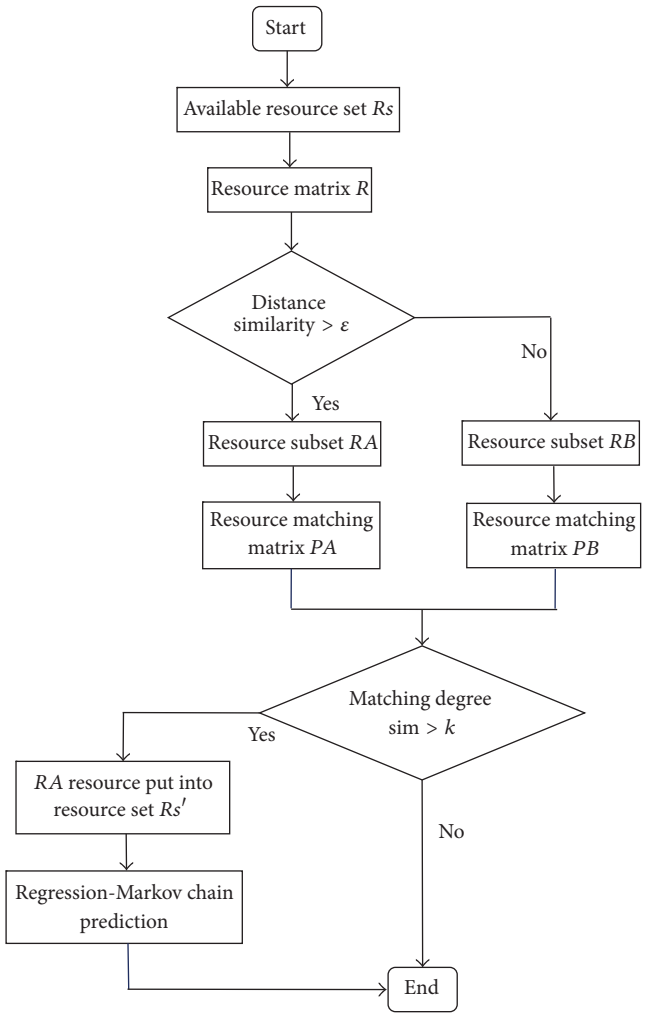

Figure 3: Flow chart of resource estimation method.

Each resource has different QoS attributes. According to these attributes, the resource is evaluated to determine the matching between the needs of users and resources. Resource set Rq_i denotes a collection of the QoS attributes of resource.

$$Rq_i = \{rq_1, rq_2, \ldots, rq_m\}, \tag{2}$$

where index m represents that each resource has m different QoS attributes, including response time, availability, reliability, and price. The price attribute in this paper is defined as follows:

$$p = U * b\frac{\mu}{\varphi} * D_{\text{dev}}, \tag{3}$$

where U is the basic price of service. μ is the number of service requests completed in unit time. φ is the number of service requests received in unit time. b is a price adjustment factor, determined by the service provider. D_{dev} represents the device type, which can generally be divided into static devices, small mobile devices, and large mobile devices. The relative reserved resources of each device are 1, 1.25, and 1.5 times, respectively [10].

The QoS attributes requested by the users are the same as the resources. The QoS attributes set is defined as follows:

$$uq = \{uq_1, uq_2, \ldots, uq_m\}. \tag{4}$$

In this paper, we construct a QoS attribute matrix of resources as the decision matrix.

$$R = \left(r_{ij}\right)_{n\times m} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{pmatrix}, \tag{5}$$

where r_{ij} is the jth QoS attribute value of ith resource.

Since the measurement units of the QoS attributes are different, it is meaningless to process the matrix directly. Therefore, according to the relationship between attributes and user satisfaction, we use the following formula to standardize processing:

$$z_{ij} = \begin{cases} \dfrac{r_{ij} - r_j^{\min}}{r_j^{\max} - r_j^{\min}}, & q \text{ is positive attribute} \\ \dfrac{r_j^{\max} - r_{ij}}{r_j^{\max} - r_j^{\min}}, & q \text{ is negative attribute.} \end{cases} \tag{6}$$

The above formula indicates two cases: if QoS attribute q is positive attribute, the greater the value of the attribute is, the higher the satisfaction of the users is. On the contrary, the negative attribute indicates that the smaller the attribute value is, the higher the user satisfaction is.

In order to ensure the objectivity of the estimation results, we use the entropy weight method to calculate entropy value e_j and entropy weight w_j for the QoS attribute of resources. The formula is as follows:

$$\begin{aligned} e_j &= -\frac{1}{\ln n}\sum_{i=1}^{n} f_{ij} \cdot \ln f_{ij} \\ w_j &= \frac{1 - e_j}{m - \sum_{j=1}^{m} e_j} \\ f_{ij} &= \frac{z_{ij}}{\sum_{i=1}^{n} z_{ij}} \\ \sum_{j=1}^{m} w_j &= 1. \end{aligned} \tag{7}$$

4.1.2. Similarity Matching Algorithm. In order to reduce the matching time and improve the matching efficiency, the resources are firstly classified before similarity matching. Based on the QoS attributes value requested by the users, each resource can be regarded as a point in the multidimensional space. The distance between the resources and the needs of users is measured by Euclidean distance. Since each user may have preferences for one attribute, or each attribute of the resources affects the result of the measurement differently,

therefore, we set objective weight for each attribute and use weighted method to calculate.

$$d(i, uq) = \sqrt{\sum_{j=1}^{m} w_j * \left(rq_j - uq_j\right)^2}$$
$$d_{\text{sim}}(i, uq) = \frac{1}{1 + d(i, uq)}, \tag{8}$$

where w_j represents the weight of resource attribute. $d(i, uq)$ indicates the distance between the ith resource node and the ideal node (the node indicates the QoS requirements of users) in the space. $d_{\text{sim}}(i, uq)$ is the degree of proximity, whose range is from 0 to 1. The resource is close to the ideal node when $d(i, uq)$ is smaller.

We set the proximity threshold ε by computing the proximity degree between the resource node and the ideal node. The range of threshold is from 0 to 1. It divides the resources into two sets.

$$R'(\varepsilon) = \left\{r_k \mid d_{\text{sim}}(k, uq) \geq \varepsilon\right\}. \tag{9}$$

On the basis of the resource classification, we establish the matching matrix between the requested QoS of users and the QoS of the candidate resources.

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n+1)1} & p_{(n+1)2} & \cdots & p_{(n+1)m} \end{pmatrix}, \tag{10}$$

where the first n row represents the QoS attributes value of the n resources. The $n + 1$ row represents the QoS attribute value of the expectation of users. We use formula (6) to standardize the matrix and calculate the matching degree between the expectation resource of users and the resources.

In this paper, we use Pearson correlation coefficient to calculate the matching degree of resources.

$$\text{sim}(x, y) = \frac{\Sigma_{r \in R}\left(q_{r,x} - \overline{q}_x\right)\left(q_{r,y} - \overline{q}_y\right)}{\sqrt{\Sigma_{r \in R}\left(q_{r,x} - \overline{q}_x\right)^2}\sqrt{\Sigma_{r \in R}\left(q_{r,y} - \overline{q}_y\right)^2}}, \tag{11}$$

where $\text{sim}(x, y)$ is the similarity between attributes x and y. $q_{r,x}$ and $q_{r,y}$ represent the corresponding value of the QoS attributes x and y of all resources, respectively. $\overline{q}_x$ and $\overline{q}_y$ represent the average value of attributes x and y of all resources, respectively.

In order to ensure that the QoS attribute value of candidate resources is within the expected range of users, we introduce the penalty factor and Grey incidence matrix to analyze the gap between the resource attribute value. The penalty factor λ is set on the condition of resource constraint. The smaller the penalty factor is, the closer the user expectation range is, and the higher the correlation is. On the contrary, when the penalty factor is larger, the distance is more away from the expectation range of users, the lower the correlation is, and the lower the matching degree is.

$$\lambda_j = \begin{cases} 0 & r_j < \delta_j \\ \dfrac{r_j - \delta_j}{\gamma_j - \delta_j} & \delta_j \leq r_j \leq \gamma_j \\ 1 & r_j > \gamma_j, \end{cases} \tag{12}$$

where δ_j and γ_j represent corresponding minimum value and maximum value of the expectation range of users for each attribute of the resources, respectively.

We introduce the penalty factor into the Grey relational coefficient to calculate correlation degree and correct matching function. The attribute correlation coefficient is calculated as follows:

$$\xi_{ij} = \frac{\min + \eta \cdot \max}{\lambda_j \left|p_{n+1}(j) - p_i(j)\right| + \eta \cdot \max}$$
$$\min = \min \min \left|p_{n+1}(j) - p_i(j)\right| \tag{13}$$
$$\max = \max \max \left|p_{n+1}(j) - p_i(j)\right|.$$

The Grey incidence matrix is defined as follows:

$$\xi = \begin{bmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1m} \\ \xi_{21} & \xi_{22} & \cdots & \xi_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \xi_{n1} & \xi_{n2} & \cdots & \xi_{nm} \end{bmatrix}. \tag{14}$$

ξ_{ij} is correlation coefficient of the jth attribute of the ith resource. η is the distinguishing coefficient, with a general value of 0.5. λ_j is a penalty factor of the jth attribute. min and max represent the maximum difference and the minimum difference. $p_i(j)$ and $p_{n+1}(j)$ are standardized values.

The modified matching function is

$$\text{sim}(r_i, uq) = \alpha * \text{sim}(rq_i, uq_i) + (1 - \alpha) * \frac{1}{m}\sum_{j=1}^{m} \xi_{ij}, \tag{15}$$

where α represents weight. When the matching degree reaches the threshold k, that is, $\text{sim}(r_i, uq) \geq k$, this indicates a successful match. The resources that successfully match are put into a candidate resource set Rs'.

The specific matching algorithm is illustrated in Algorithm 1.

4.2. Regression-Markov Chain Prediction Estimation. Through the above analysis, we select a candidate resource set which satisfy the QoS requirements of users. Because of the dynamic and uncertain nature of the resources, the amount and value of data collected of QoS will fluctuate. Moreover, the inherent mobility of the IoT makes mobile users have certain volatility when using resources. Therefore, we further select resources by analyzing the load changes of resources themselves.

Input: $Rs = \{r_1, r_2, \ldots, r_n\}, R, uq$
Output: Rs'
(1) Normalized matrix R
(2) $[n, m] = \text{size}(R)$
(3) **for** $j = 1 \rightarrow m$ **do**
(4) get the w_j
(5) **end for**
(6) $Q = [R; uq]$
(7) $[\text{row}, \text{col}] = \text{size}(Q)$
(8) $RA[] \leftarrow 0$
(9) $RB[] \leftarrow 0$
(10) $u \leftarrow 1$
(11) $v \leftarrow 1$
(12) **for** $i = 1 \rightarrow$ row-1 **do**
(13) get the $d(i, uq)$ and $d_{\text{sim}}(i, uq)$
(14) **if** $d_{\text{sim}}(i, uq) \geq \varepsilon$ **then**
(15) $RA[u] \leftarrow r_i$
(16) $u \leftarrow u + 1$
(17) **else**
(18) $RB[v] \leftarrow r_i$
(19) $v \leftarrow v + 1$
(20) **end if**
(21) **end for**
(22) $Rs'[] \leftarrow 0$
(23) $t \leftarrow 1$
(24) **while** resource $r_i \in RA$ **do**
(25) get the matrix PA
(26) calculate the λ_j and ξ_{ij}
(27) calculate the $\text{sim}(r_i, uq)$
(28) **if** $\text{sim}(r_i, uq) \geq k$ **then**
(29) $Rs'[] \leftarrow r_i$
(30) $t \leftarrow t + 1$
(31) **end if**
(32) **end while**
(33) **return** Rs'

Algorithm 1: Multiattribute QoS resource matching.

Due to the randomness and volatility of the load, the single resource prediction method is difficult to predict accurately. The load has the characteristics of frequent changes in the short term [18]. The future load will be affected by the current load, and the load at the next time can be predicted by current load value. Therefore, this paper adopts the regression-Markov chain prediction method to better reflect the changing trend and stochastic fluctuation characteristics of resources.

According to the difference of time t, the original data sequence of the resource load value is recorded as

$$\left\{X^{(0)}(t)\right\} = \left\{X^{(0)}(1), X^{(0)}(2), \ldots, X^{(0)}(p)\right\}. \quad (16)$$

Firstly, we use the linear regression method to generate the predicted sequence

$$\left\{\widehat{X}^{(0)}(t)\right\} = \left\{\widehat{X}^{(0)}(1), \widehat{X}^{(0)}(2), \ldots, \widehat{X}^{(0)}(p)\right\} \quad (17)$$

and calculate the relative residuals of the original sequence and the predicted sequence

$$\left\{X^{(1)}(t)\right\} = \left\{\widehat{X}^{(0)}(t) - X^{(0)}(t)\right\}. \quad (18)$$

Then, we divide the relative residual sequence $\{X^{(1)}(t)\}$ into s state intervals. According to the state distribution $S = (1, 2, \ldots, s)$, a one-step transition probability matrix is calculated.

$$M_{s\times s} = M(s_i)(s_j) = p_{ij}(X_{s+1} = j \mid X_s = i)$$

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1s} \\ M_{21} & M_{22} & \cdots & M_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ M_{s1} & M_{s2} & \cdots & M_{ss} \end{bmatrix}. \quad (19)$$

If matrix M satisfies the following conditions: $M_{ij} \geq 0$, $i, j \in S$, and $\sum M_{ij} = 1$, $i, j \in S$, M is a random matrix. $P^{(n)} = (p_{ij}^{(n)}) = P^n$ represents n step transfer matrix.

It is assumed that the Markov chain reaches a stable state after the n step transition. The stable state vector $x = [x_1, x_2, \ldots, x_s]$ satisfies

$$x = xM$$
$$\sum_{i=1}^{s} x_i = 1. \quad (20)$$

We set the initial state to obtain the probabilities $p_1, p_2, \ldots, p_s$ corresponding to the error state interval by solving the distribution probability of the stable state. According to the maximum probability principle, the state corresponding to the maximum probability is taken as the state of the next moment. The predictive value is brought into the corresponding state interval to solve the prediction interval value. On this basis, we predict the availability of resources over a period of time and rationally select the right resource.

5. Experimental Results

To test the performance of estimation method, we use the performance indicators in [13] as the estimation indicators. The performance indicators in this experiment are defined as follows.

Precision (Prec = $|A|/|B|$) is used to measure the accuracy of resource selection. Specifically, candidate resources that successfully match the user QoS make up the percentage of all resources, where A represents candidate resources that successfully match the user QoS and B represents all the resources.

Recall (Rec = $|A|/|A \cup C|$) is used to measure the effectiveness of resource selection. Specifically, the candidate resources that successfully match the user account for the percentage of all matching successful resources, where C represents the matching successful resources that are not within the candidate resources set.

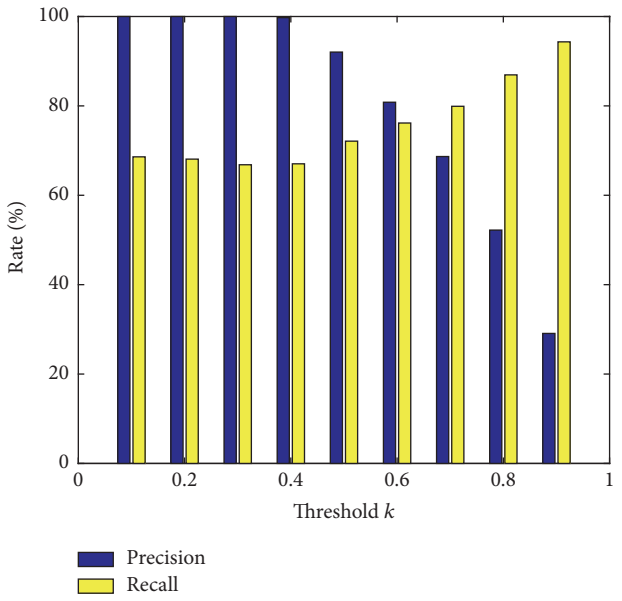

FIGURE 4: Variation of precision and recall under different matching threshold.

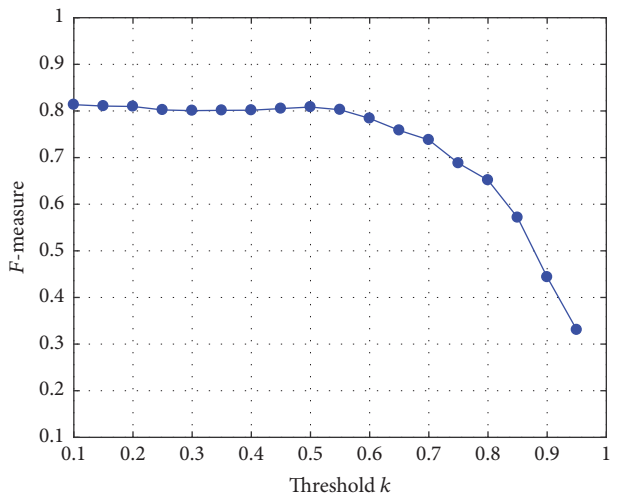

FIGURE 5: Variation of F-measure under different matching threshold.

F-measure (F-measure = $(2 * \text{Prec} * \text{Rec})/(\text{Prec} + \text{Rec})$) is the weighted average of precision and recall, which is used to reflect the overall performance. The more the F-measure tends to 1, the more effective the estimation method is.

The experimental platform adopts MATLAB to set up a different number of resource nodes randomly. Each resource node includes multiple QoS attribute information. The QoS attributes information is set by simulation of resources characteristics on edge servers, including response time, availability, and price. The proximity degree ε and the weight α are 0.5. In order to ensure the effectiveness of the result, we conduct the experiment 20 times and take the average value as the experimental result.

In order to obtain the best performance, we set up a matching threshold k to filter irrelevant resources by low similarity. Taking into account the contradictory relation between precision and recall, we decide to use F-measure as the initial standard to find the optimal threshold.

Figure 4 shows the variation of precision and recall under different matching threshold. As can be seen from Figure 4, the precision and recall are inversely related. When the matching degree is higher, the precision is lower, and the recall is higher. In order to ensure the precision and recall within acceptable limits, the threshold k should be between 0.5 and 0.6. Figure 5 shows the change of F-measure under different matching threshold. It can be seen that, with the increase of threshold, the F-measure is significantly decreased. When the k value is 0.5, the F-measure is in the higher position, so $k = 0.5$ is taken as the matching threshold.

Figures 6 and 7 show the precision and recall performance of the method, respectively. Our method is compared with two methods where the first does not consider the penalty factor (denoted as SMNP method) and the second does not consider the penalty factor and Grey relational grade

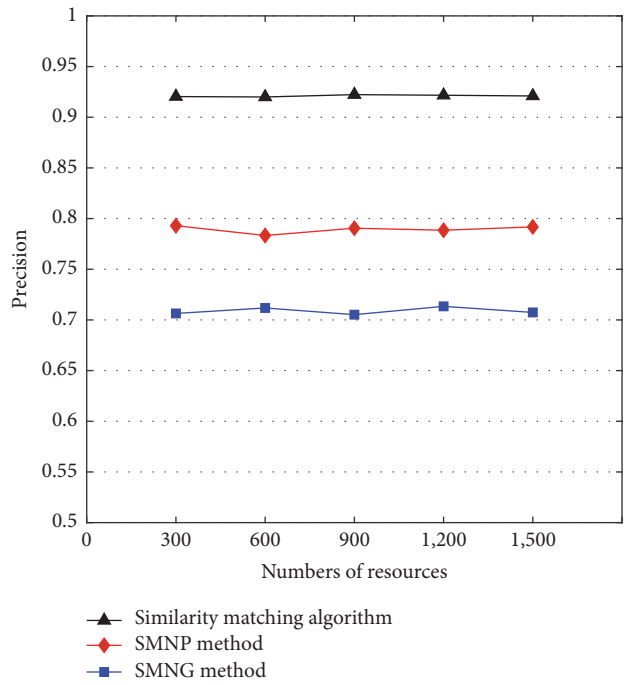

FIGURE 6: The comparison of precision of different methods under different numbers of resources.

(denoted as SMNG method), respectively. As we can see, our method is superior to other methods on precision and stays above 90%. Since the penalty factor improves the similarity of resources, it makes the resources meet the needs of users as much as possible. But the recall is lower than the other two methods and remains at around 72%, within the acceptable level. Although the method can avoid the constraint and relationship between the attributes of resources, it only matches the quantity attribute of the resources.

Figure 8 shows the F-measure under the different numbers of resources. It can be seen that our method is superior

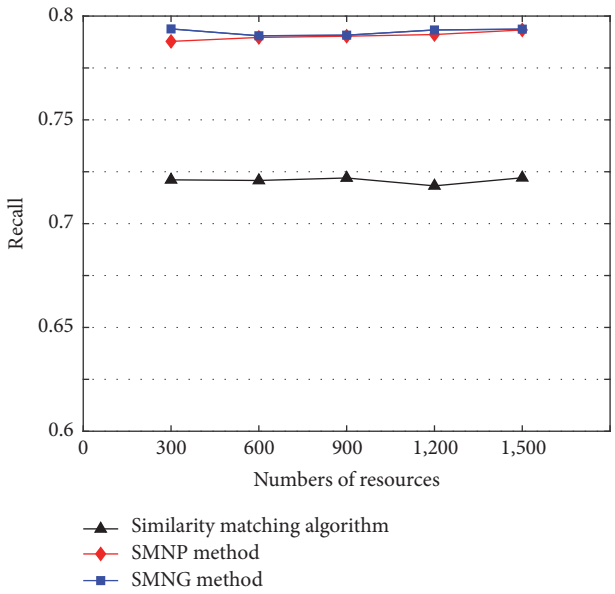

Figure 7: The comparison of recall of different methods under different numbers of resources.

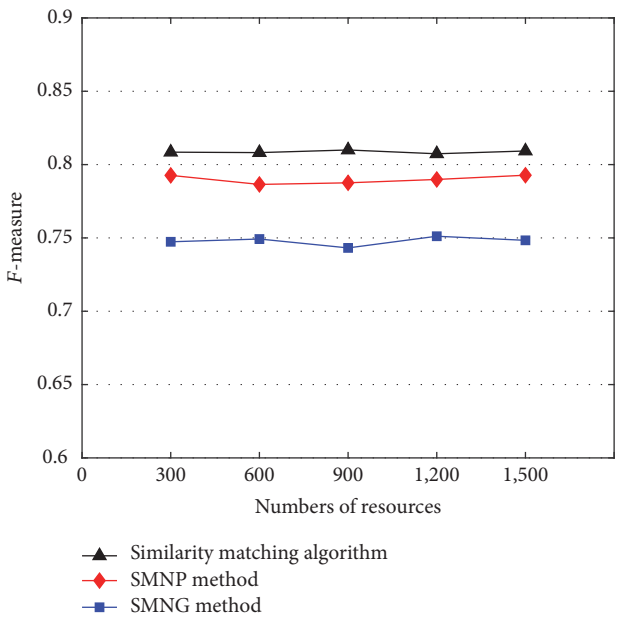

Figure 8: The comparison of F-measure of different methods under different numbers of resources.

Table 1: Residual state interval.

State	Residual interval
(1)	(−20%, −10%]
(2)	(−10%, −5%]
(3)	(−5%, 0%]
(4)	(0%, 5%]
(5)	(5%, 10%]
(6)	(10%, 20%]

to other methods. The F-measure is maintained above 80%, which further validates the effectiveness of our method.

In this paper, we take CPU utilization as resource load index to predict. We record CPU utilization of downloading files every 5 seconds to obtain time series data. We use the first 10 times' load data to make short-term prediction of the next 5 times' CPU utilization and prove the effectiveness of the method by the error values. The residual sequence is divided into the following 6 states by linear regression analysis, as shown in Table 1.

As the regression-Markov chain prediction method is interval prediction method, the prediction result is interval distribution. We select the state interval with the maximum probability as the prediction interval. We select (−5%, 0%] state interval as a result by analyzing the probability of each state interval in the stable state.

The error result of the regression-Markov chain prediction method is shown in Figure 9. It can be seen that the method has a higher prediction accuracy and less error compared with the linear regression prediction method. Due to the large fluctuation and randomness of the load, the single prediction method has a poor effect. The linear regression method can predict the changing trend of the load state, but it can not reflect the stochastic volatility. The Markov chain solves the stochastic volatility problem and improves the accuracy of the prediction method.

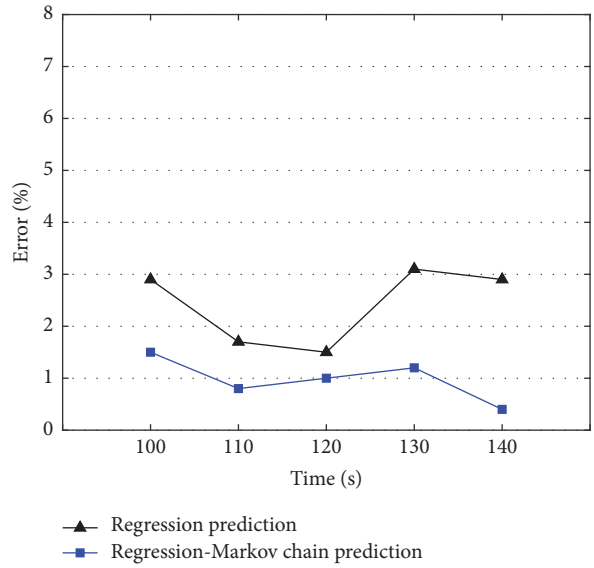

Figure 9: Comparison of errors between regression prediction and regression-Markov chain prediction in the future time.

6. Conclusion and Future Work

With the rapid development of Internet of Things and cloud computing, service QoS and user satisfaction become an important challenge. Local computing and storage capabilities of edge computing can reduce latency and improve user satisfaction. In this paper, we use weighted Euclidean distance similarity to classify multiple QoS attribute resources. We

select the appropriate resources by similarity matching and regression-Markov chain prediction method. Since the QoS attributes system is extensible and the user QoS requirements are dynamic, the estimation method has certain scalability. On the basis of the existing work, we can design a reasonable method of resource estimation to balance the satisfaction between users and service providers and improve the utilization of resources.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61672321, 61771289), the Shandong provincial Postgraduate Education Innovation Program (SDYY14052, SDYY15049), the Shandong Provincial Specialized Degree Postgraduate Teaching Case Library Construction Program, the Shandong Provincial Postgraduate Education Quality Curriculum Construction Program, the Shandong Provincial University Science and Technology Plan Project (J16LN15), and the Qufu Normal University Science and Technology Plan Project (xkj201525).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] D. Evans, *The Internet of Things: How the Next Evolution of the Internet is Changing Everything*, Cisco Systems, 2011.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the Mcc Workshop on Mobile Cloud Computing*, pp. 13–16, ACM, August 2012.

[5] S. M. Hemam and O. Hioual, "Load balancing issue in cloud services selection by using MCDA and Markov Chain Model approaches," in *Proceedings of the International Conference on Cloud Computing Technologies and Applications*, pp. 163–169, IEEE, May 2016.

[6] M. M. Al-Sayed, S. Khattab, and F. A. Omara, "Prediction mechanisms for monitoring state of cloud resources using Markov chain model," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 163–171, 2016.

[7] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[8] S. Shekhar and A. Gokhale, "Dynamic resource management across cloud-edge resources for performance-sensitive applications," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 707–710, IEEE Press, May 2017.

[9] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: a framework for edge node resource management," *IEEE Transactions on Services Computing*, 2017.

[10] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications (AINA '15)*, vol. 51, no. 5, pp. 687–694, IEEE Computer Society, March 2015.

[11] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT," in *Proceedings of the 23rd International Conference on Telecommunications (ICT '16)*, pp. 1–5, May 2016.

[12] L.-Y. Zuo, Z.-B. Cao, and S.-B. Dong, "Virtual resource evaluation model based on entropy optimized and dynamic weighted in cloud computing," *Journal of Software*, vol. 24, no. 8, pp. 1937–1946, 2013.

[13] S. S. Zhao, Y. Zhang, L. Yu, B. Cheng, Y. Ji, and J. Chen, "A multidimensional resource model for dynamic resource matching in internet of things," *Concurrency & Computation Practice & Experience*, vol. 27, no. 8, pp. 1819–1843, 2015.

[14] J. Zhou, M. Yan, X. Ye, and H. Lu, "An algorithm of resource evaluation and selection based on Multi-QoS constraints," in *Proceedings of the Web Information Systems & Applications Conference*, pp. 49–52, IEEE Computer Society, 2010.

[15] W. E. Dong, W. U. Nan, and L. I. Xu, "QoS-oriented monitoring model of cloud computing resources availability," in *Proceedings of the 5th International Conference on Computational and Information Sciences*, pp. 1537–1540, June 2013.

[16] S. Ding, C. Xia, Q. Cai, K. Zhou, and S. Yang, "QoS-aware resource matching and recommendation for cloud computing systems," *Applied Mathematics and Computation*, vol. 247, no. 15, pp. 941–950, 2014.

[17] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, 2017.

[18] L. Zhang and D. Y. Xu, "Multi-step optimized GM (1, 1) model-based short term resource load prediction in cloud computing," *Computer Engineering and Applications*, vol. 50, no. 10, pp. 65–71, 2014.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7172045, 6 pages
https://doi.org/10.1155/2018/7172045

Research Article

Cloud/Fog Computing System Architecture and Key Technologies for South-North Water Transfer Project Safety

Yaoling Fan,[1] **Qiliang Zhu**,[2] **and Yang Liu**[2]

[1]*School of Water Conservancy & Environment, Zhengzhou University, Zhengzhou, China*
[2]*School of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou, China*

Correspondence should be addressed to Yaoling Fan; fyl@ncwu.edu.cn

Received 4 June 2017; Accepted 17 September 2017; Published 16 January 2018

Academic Editor: Shangguang Wang

In view of the real-time and distributed features of Internet of Things (IoT) safety system in water conservancy engineering, this study proposed a new safety system architecture for water conservancy engineering based on cloud/fog computing and put forward a method of data reliability detection for the false alarm caused by false abnormal data from the bottom sensors. Designed for the South-North Water Transfer Project (SNWTP), the architecture integrated project safety, water quality safety, and human safety. Using IoT devices, fog computing layer was constructed between cloud server and safety detection devices in water conservancy projects. Technologies such as real-time sensing, intelligent processing, and information interconnection were developed. Therefore, accurate forecasting, accurate positioning, and efficient management were implemented as required by safety prevention of the SNWTP, and safety protection of water conservancy projects was effectively improved, and intelligential water conservancy engineering was developed.

1. Background

Water conservancy engineering presents the national fundamental industry and is vital in national economic development. The construction of water conservancy projects is generally large in scale, with high investment, wide geographical distribution, and decentralized management, in a long construction period and contains a large amount of information. It may be also affected by unfavorable terrain conditions, complex geological structure, and flood, which increase the difficulty in supervision. Therefore, the construction of Internet safety information system is of great significance for improving safety monitoring and ensuring project safety and water quality safety in water conservancy engineering.

With the rapid development of Internet of Things (IoT) technology in recent years, an important source of big data is the tremendous amount of data that are collected, transferred, stored, and processed through the IoT system [1–3]. Due to the general dense geographical distribution of water conservancy projects, the IoT-based data in water conservancy engineering have typical geographical attributes.

Taking the Middle Route of the SNWTP as an example, its main line starts at Taocha Sluice of Danjiangkou Reservoir, crosses the Yangtze River, Huaihe River, Yellow River, and Haihe River, passes through Provinces of Henan, Hebei, Beijing and Tianjin, and provides water for tens of large- and medium-sized cities, like Pingdingshan, Xuchang, Zhengzhou, Jiaozuo, Xinxiang, Hebi, Anyang, Handan, Xingtai, Shijiazhuang, Baoding, Beijing, Tianjin, and so on. Therefore, it is necessary to build the safety information network system for the water conservancy project.

Combined with the research work of the National Major Project "Research and Development of Sensor Network Technology Oriented to the Safety of the South-North Water Transfer Project (SNWTP)," this study investigated the cloud/fog network system architecture for project safety of the Middle Route of SNWTP. This paper first analyzed the application and problems of cloud computing in the water conservancy industry and then proposed the cloud/fog network architecture for project safety of the Middle Route of SNWTP. It is also responsible for the occurrence of false anomaly data from the bottom sensors of the water

conservancy project network; a method of data reliability detection for the false alarm is put forward.

2. Application of Cloud Computing in Water Conservancy Industry

Cloud computing, as a new generation of computing architecture with strong scalability, has become a basic platform to support big data applications, and it effectively satisfies the growing data processing and storage requirements in areas such as water conservancy engineering [4–6]. In terms of service pattern, Sun et al. designed and implemented the hydrological simulation and water resources management in the cloud computing model for the specific application requirements of distributed hydrological modeling and water resources monitoring [7, 8]. Ari and Muhtaroglu designed the cloud service framework of finite element simulation and implemented some modules [9]. Zhang and Wang proposed the technology architecture of an integrated management information platform in water conservancy engineering based on cloud computing technology [10]. R. J. Yang and Y. J. Yang set up an experimental platform for water resources management system based on private cloud service [11]. Zhou et al. studied the cloud computing platform system and its application in water resources research [12].

The aforementioned studies provide technical support and implementation ideas for solving some water conservancy problems, especially in terms of efficient use of resources, scalability, and cost savings. However, the studies also demonstrated the limitations of the cloud computing architecture. In the application of big data of IoT in water conservancy engineering, especially due to high requirement for latency and dense geographical distribution of the water resources data, the problems are presented as follows.

(1) It is unable to meet the requirement of low latency in early warning applications such as water quality safety and human safety.

Traditional cloud computing causes large round-trip delays because the enormous amounts of data generated from the distributed sources have to be transferred to the computing center for processing. The situation will be more severe when the amount of data and transferring speed exceed the current capacity of the equipment, so it may take several days or weeks to transfer terabytes of data to the cloud and bring huge administrative costs.

(2) The enormous amounts of water conservancy engineering data are geographically distributed.

Water conservancy projects generally cover a large area. Taking the Middle Route of the SNWTP as an example, there is a management station in each bypassed city, and each management station has a number of sluice gate stations. The sensors of water conservancy facilities are widely distributed, so that the centralized cloud computing method is not suitable.

(3) The network of IoT is generally complex, and it does always meet the requirements of transmission bandwidth and reliability in cloud computing.

The computing power of the facilities at sluice stations is limited, largely restricted by the energy supply in the field and poor onsite installation environment, so the bandwidth and reliability of the formed network architecture are far from satisfying the requirements of traditional cloud computing.

3. Characteristics of Fog Computing Architecture

In order to take advantage of the flexibility of cloud computing architecture as well as to solve the above problems encountered in the IoT big data applications, Cisco Inc. proposed the concept of fog computing in 2012.

Fog computing is conceptually the extension of cloud computing. Fog computing was named after cloud computing due to the concept that "cloud at ground level is known as fog." Compared with cloud computing, the architecture used for fog calculation is more distributed and closer to network edges. Fog computing arranges data, data processing, and applications at network edges, which is unlike cloud computing that keeps them almost entirely in the cloud. The storage and processing of data are more reliable on local facilities than servers in fog computing. Therefore, cloud computing is a new generation of centralized computing, while fog computing is a new generation of distributed computing, which is in line with the "decentralization" feature of Internet [13, 14].

Fog computing is mainly based on small cloud such as personal, private, and enterprise cloud, while cloud computing is mainly based on IT services, public cloud. Fog calculation is powerful in large quantity and emphasizes the quantity, where the single computing node plays an important role, while cloud computing emphasizes the overall computing power, which is typically calculated by a bunch of concentrated high-performance computing facilities. Fog calculation expands the network-computing model of cloud computing and extends the network computing from the center to the edges of the network, as shown in Figure 1, which is therefore more widely used in a variety of applications [15–18].

The noticeable characteristics of fog calculation are as follows.

(1) Low latency and position sensing characteristics as fog computing locates at edges: it is of significance to the current informatics IoT in water conservancy engineering; for example, low latency is required in sudden floods, water pollution, and personal safety.

(2) Dense geographical distribution: this coincides with the wide distribution of water management of the SNWTP.

(3) Enormous amounts of nodes: a large-scale sensor network with numerous network nodes can be used to monitor the environment.

(4) Adaptability to access to mobile devices: mobile devices in fog computing can communicate directly with each other without transferring to the cloud or station, so that fog computing is highly adaptable to mobile devices, for example, mobile water quality monitoring and water level monitoring [19].

(5) High real-time feature: fog computing supports computation and processing of data at network edges with a low latency [20].

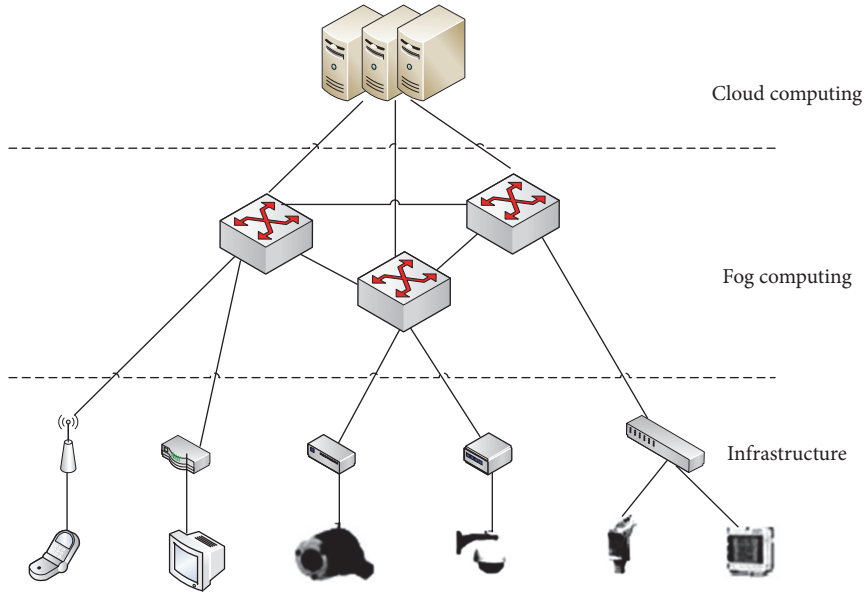

Figure 1: Fog between edge and cloud.

(6) Supportive to diverse heterogeneous software and hardware: fog computing devices are inherently heterogeneous and are deployed in different locations in the environment, such as cores, edges, access networks, and endpoints.

4. Safety System Architecture for the SNWTP

4.1. Monitoring Object of the SNWTP. The SNWTP is a strategic project in China. It is the infrastructure to optimize water resources allocation and promote regional coordinated development. Therefore, it is extremely important to ensure project safety, water supply safety, and human safety. Taking the Middle Route of the SNWTP as an example, the specific attributes of the three types of safety monitoring are as follows.

4.1.1. Project Safety Monitoring. The main line of the Middle Route is 1432 km in length, including 318 control buildings, 1256 bridges, and 469 left bank drainage structures. Therefore, it is necessary to prevent and control safety problems due to channel seepage or leakage, seepage in inverted siphons or dark culverts, and uneven settlement or deformation of buildings. The monitoring task is challenging.

4.1.2. Water Supply Safety Monitoring. For the Middle Route of the SNWTP, it requires a Class II water quality when the water is transferred to Beijing. However, it is difficult to ensure the water quality at the source region and to control the point and nonpoint pollution. In addition, sudden accidents easily cause water pollution. The quantity safety level is also high. Therefore, high accuracy is required for real-time gates operation, overall water surface profile control, and water level and flow monitoring. A wide range of meteorological and hydrological monitoring is also needed.

4.1.3. Human Safety Monitoring. Because the Middle Route of the SNWTP extensively crosses over local highways, railways, and rivers, and open management strategies are adapted; therefore, the risk of external invasion is high. When outsiders enter the channel to play with water or go fishing, hidden dangers exist, for example, dawning of people or stealing of important facilities, which may result in water conveyance accidents and threat human safety. Rigorous monitoring of human safety is necessary.

4.2. A Cloud/Fog Network System Architecture. There are more than 5,000 wired or wireless sensors in the Middle Route of SNWTP, with high-definition camera devices installed at every 500 m. These basic data acquisition devices and related computing resources and network resources represent the IoT nodes of the Middle Route of SNWTP. Based on the enormous amount of data generated by IoT, relevant big data analysis is conducted to apply in the safety prevention of water conservancy projects and to solve problems that are not able to or difficult to be solved by traditional methods. This study presents a cloud/fog network system architecture, as shown in Figure 2, oriented to the safety of the Middle Route of the SNWTP in water conservancy engineering.

4.2.1. Infrastructure Layer. This layer is the information input point of the water conservancy project safety platform, including various types of sensors and some network physical devices for water conservancy data acquisition. The sensors are deployed in the scattered geographical areas according to the application needs. Real-time monitoring of the corresponding water conservancy projects is implemented by sensor network, IoT technology, remote sensing technology, video capture technology, and so on. The sensors are mainly responsible for detection and collection of basic data such as osmotic pressure, water level, water flow, and precipitation.

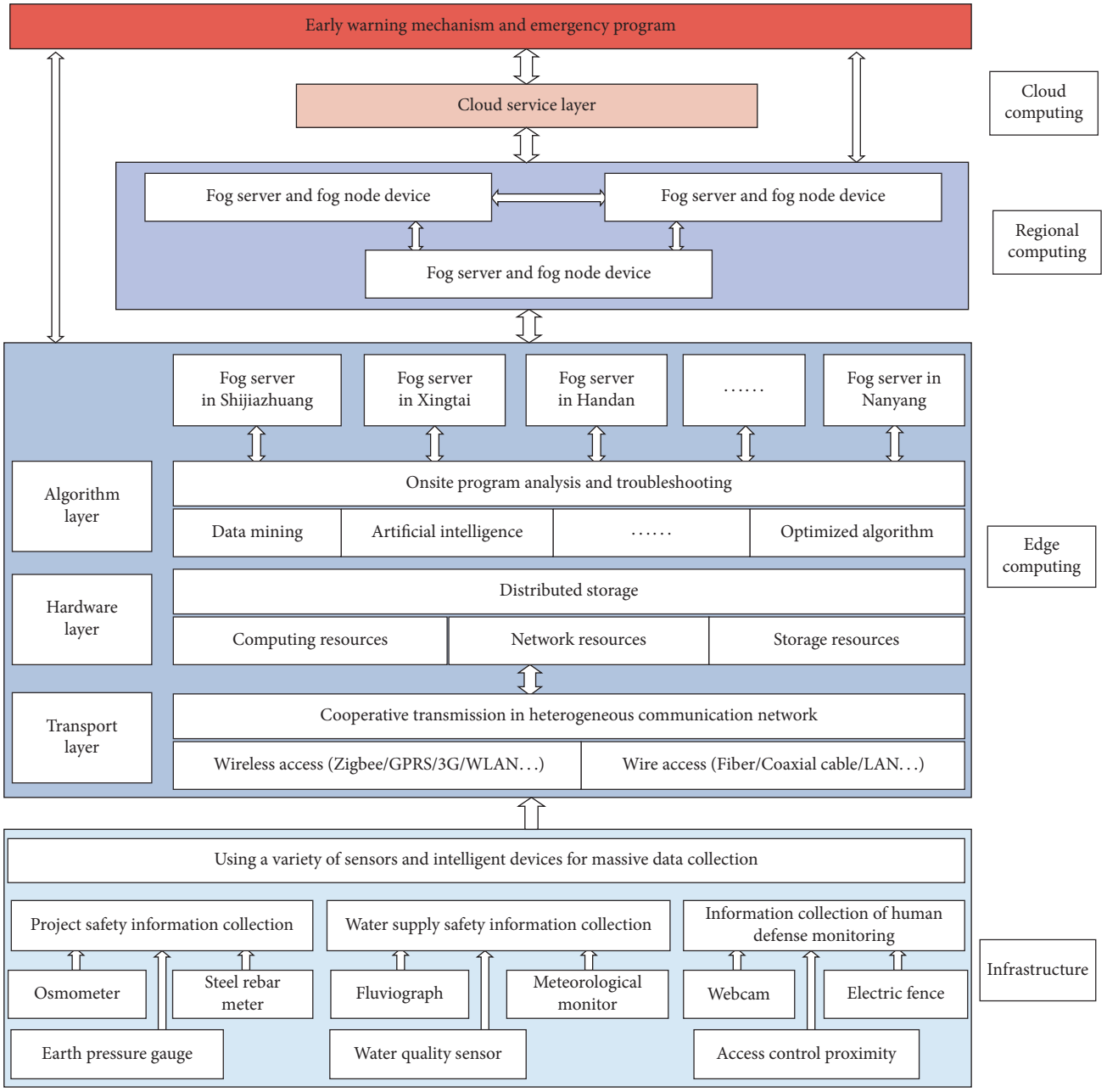

Figure 2: The cloud/fog network system architecture for project safety of the Middle Route of SNWTP.

The data are transferred to the jurisdictional water management center based on the IoT agreement in the form of data flow.

4.2.2. Edge Computing Layer. This layer is used for data storage, processing, and analysis at each water conservancy management center, which belongs to the computing edge unit in the whole network. The layer is divided into hardware layer and virtualization layer, including the corresponding hardware and software resources like network fog server, operating system, storage, database management, and so on. Among them, the fog server collects the data from the data collecting devices at water conservancy project sites, followed by cleaning, filtering, and fusion of the collected data.

The local computing mainly uses different data processing software and a variety of data analysis algorithms to process and analyze the real-time monitoring data stored at local hardware, in order to implement onsite low-latency real-time monitoring. Due to the large amount of safety monitoring data in the Middle Route of the SNWTP, the edge computing equipment can balance the load by distributed calculation of data from each region. Results used as a reference for management center can be obtained in a relatively short time; therefore, quick decision can be made and the data processing and transmission waiting time is largely reduced.

4.2.3. Regional Management Layer. This layer can manage and compute the local data at each regional management center, acquire and share classified data of each region at the edge computing units, and implement the middle level computation. For example, the fog server at this layer can track, perform analysis, and forecast based on the abnormal water quality data of each region. The computational load and the time delay to the management center are relatively larger than those at the edges.

4.2.4. Cloud Computing Layer. Cloud computing allows a large number of computing tasks to share high-speed hardware resources through the establishment of large-scale computing center and virtualization technology, which can effectively reduce computing and hardware maintenance costs. The layer focuses more on the application of high latency data with a large number of data types and complex computational model. It is used for safety risk assessment for the entire SNWTP and provides a computation platform for large regional intelligential water conservancy projects.

4.2.5. Early Warning Mechanism and Emergency Program. The early warnings of different level respond to different contingency plans. Optimized early warning program for safety of the Middle Route of SNWTP is developed, and early warning for client is constructed. The Android or Apple phone and computer are used as the early warning release end.

The cloud/fog network system architecture for project safety of the Middle Route of SNWTP adapts to the characteristics of wide geographical distribution and long-distance water supply monitoring in the project, which satisfies the real-time local computing of each water conservancy management center and regional management requirements, and provides a platform of intelligent management and safety risk assessment for the entire SNWTP. This architecture is adapted to multilevel requirements, achieves a unified deployment, and better meets the needs of security monitoring of the Middle Route of SNWTP.

5. Key Technologies

The data from the bottom sensors has the characteristics of large amount and various types in the edge computing and region computing. Once the data from some sensors is pseudo-abnormal, it is bound to cause false alarm. In order to find out the real abnormal data from a large number of data and reduce false alarm rate, the data should be authenticated and analyzed.

The monitoring data of the Middle Route of the SNWTP is mainly from wired or wireless multitype sensors. The security detection mainly analyzes the security of the detected object based on alarm threshold. To reduce the false alarm frequency caused by the pseudo-anomaly data, it needs to evaluate the reliability of the collected data. The main line of the Middle Route of the SNWTP is divided into several sections according to the geographical scope; the reliability of the data is judged by analyzing the relationship between similar and heterogeneous sensors in each section.

(1) For the same type of sensors in the same section, the data of the physical nodes in the sensor network are evaluated by the confidence level of the interval estimation based on the normal historical data. To analyze the linear/nonlinear correlation between the same types of sensors and set up the correlation model, when a sensor data is abnormal, it can be predicted by other similar sensors which have strong correlation with current sensor, which determines whether the current abnormal data is caused by a broken sensor or other reasons. The above data processing improves the reliability of the data with automatic detection.

(2) For the different types of sensors in the same section, when a sensor data is abnormal, strongly related similar or heterogeneous sensors can predict whether the data is correct by analyzing the correlation and trend of the data from many types of sensors and establishing the linear or nonlinear models among heterogeneous sensors, which improves the reliability of the warning data.

6. Conclusions

Due to the limitations of the bandwidth of network and coverage area of wireless network, the cloud computing framework fails to meet the requirements of real-time systems with high reliability. Fog computing transfers the computational tasks as much as possible to the fog servers deployed onsite, which not only reduces the overall time latency, but also provides computing services in an environment without an Internet connection. This work built the cloud/fog computing architecture for safety monitoring in water conservancy engineering, which reduced the limitation of high latency and high reliability constraints based on the cloud computing architecture. The IoT big data in water conservancy engineering with dense geographical distribution were applied based on multilevel requirements and integrated to a resource integration platform for deployment. It not only better meets the requirements of water conservancy project safety application, but also is more convenient for the development and deployment.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Major Project "Research and Development of Sensor Network Technology Oriented to the Safety of the South-North Water Transfer Project (SNWTP)" (no. 2014ZX03005001).

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in *Proceedings of the 2nd*

ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC '13), pp. 15–20, Hong Kong, China, August 2013.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] M. Fazio, N. Bessis, and M. Villari, "Advances in service-oriented and cloud computing," in *Preface of CLIoT*, vol. 58, pp. 73–75, Springer International Publishing, Berlin, Germany, 2015.

[5] A. Botta, W. De Donato, V. Persico, and A. Pescape, "On the integration of cloud computing and internet of things," in *Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 23–30, Barcelona, Spain, August 2014.

[6] T. Li and J. Han, *Cloud Computing Architecture Technologies & Practice*, Qinghua University Press, Beijing, China, 2013.

[7] A. Sun, "Enabling collaborative decision-making in watershed management using cloud-computing services," *Environmental Modeling and Software*, vol. 41, pp. 93–97, 2013.

[8] C. M. Burger, S. Kollet, J. Schumacher et al., "Introduction of a web service for cloud computing with the integrated hydrologic simulation platform ParFlow," *Computers & Geosciences*, vol. 48, pp. 334–336, 2012.

[9] I. Ari and N. Muhtaroglu, "Design and implementation of a cloud computing service for finite element analysis," *Advances in Engineering Software*, vol. 60-61, pp. 122–135, 2013.

[10] L.-X. Zhang and R.-M. Wang, "Research on the technology framework of water resources management information platfrom base on cloud computing," *Journal of the Hebei Academy of Sciences*, vol. 33, no. 2, pp. 12–16, 2016.

[11] R. J. Yang and Y. J. Yang, "Infrastructure design of water resources management system based on a private cloud computing," *Journal of Guizhou University (Natural Sciences)*, vol. 30, no. 3, pp. 109–112, 2013.

[12] L. Zhou, W. Liu, and Y. Bai, "Technological study on cloud computing system and its general design for water scientific research," *Journal of Yangtze River Scientific Research Institute*, vol. 32, no. 11, pp. 119–124, 2015.

[13] Wikipedia, Fog computing [EB/OL], 2016, https://en.Wikipedia.org/wild/Fogcomputing.

[14] W. Steiner and S. Poledna, "Fog computing as enabler for the Industrial Internet of Things," *Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 310–314, 2016.

[15] W. Fang, "A paradigm shift to fog computing from cloud computing and edge computing," *Journal of Nanjing University of Information Science and Technology: Natural Science Edition*, vol. 8, no. 5, pp. 404–414, 2016.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the 1st edition of the ACM Mobile Cloud Computing Workshop on Mobile cloud Computing (MCC '12)*, pp. 13–15, Helsinki, Finland, August 2012.

[17] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for IoT," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.

[18] M. Chiang and T. Zhang, "Fog and IoT: an overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[19] T. V. N. Rao, M. A. Khan, M. Maschendra et al., "A paradigm shift from cloud to fog computing," *IJCSET*, vol. 5, no. 11, pp. 385–389, 2015.

[20] N. Peter, "Fog computing and its real time applications," *International Journal of Emerging Technology and Advanced Engineering*, vol. 5, no. 6, pp. 266–269, 2015.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7472095, 10 pages
https://doi.org/10.1155/2018/7472095

Research Article

Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC

Xingshuo An,[1] Xianwei Zhou,[1] Xing Lü,[1] Fuhong Lin,[1] and Lei Yang[2]

[1]*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China*
[2]*Department of Computer Science and Engineering, University of Nevada, Reno, Reno, NV, USA*

Correspondence should be addressed to Fuhong Lin; fhlin@ustb.edu.cn

Received 5 September 2017; Accepted 16 November 2017; Published 14 January 2018

Academic Editor: Shangguang Wang

Fog computing, as a new paradigm, has many characteristics that are different from cloud computing. Due to the resources being limited, fog nodes/MEC hosts are vulnerable to cyberattacks. Lightweight intrusion detection system (IDS) is a key technique to solve the problem. Because extreme learning machine (ELM) has the characteristics of fast training speed and good generalization ability, we present a new lightweight IDS called sample selected extreme learning machine (SS-ELM). The reason why we propose "sample selected extreme learning machine" is that fog nodes/MEC hosts do not have the ability to store extremely large amounts of training data sets. Accordingly, they are stored, computed, and sampled by the cloud servers. Then, the selected sample is given to the fog nodes/MEC hosts for training. This design can bring down the training time and increase the detection accuracy. Experimental simulation verifies that SS-ELM performs well in intrusion detection in terms of accuracy, training time, and the receiver operating characteristic (ROC) value.

1. Introduction

With the rapid development of smart devices, we are embracing an era of the Internet of Things (IoT). The IoT applications require mobility support, geodistribution, location awareness, and low latency. However, it is difficult for cloud computing to meet these requirements. Edge paradigms such as fog computing (FC) and mobile edge computing (MEC) are proposed to overcome the above challenging issues [1–3]. The nodes, which could perform computing tasks, are called fog nodes/MEC hosts in FC and MEC hosts in MEC, which can provide low-latency service. FC and MEC are somewhat different. For example, MEC hosts are typically deployed by mobile service providers [4], and fog nodes are made up of edge servers or devices with communication and computing power. However, their network model and many features are similar [5]. They both extend cloud computing to the edge. In this work, we study a generalized network model that can be applied in FC and MEC.

Generally, the infrastructure of FC or MEC consists of three layers: the cloud server, fog node/MEC host layer, and user device layer, where fog nodes/MEC hosts are computing nodes unique to FC and closer to the user [6]. The purpose of using fog nodes/MEC hosts is to provide service with lower latency, more flexible access, and more secure network communication to network users [7].

As a new network paradigm, FC/MEC presents several challenges in network stability and performance. In FC/MEC, most of the terminal devices are resource-constrained; for example, the terminal connected to a fog node/MEC host can be a smart home appliance, a smart phone, an unmanned aerial vehicle (UAV), or a VR device [8]. Threats may come from various aspects for a network with such characteristics, such as denial of service (DoS), man in the middle (MIM), rogue gateway, privacy leakage, and service manipulation [5].

Since there are many attacks against the FC network, the benefits of FC are diminished by the damage caused by malicious attacks if no proper security and privacy protection mechanisms are available. To effectively handle security threats in FC infrastructure and to minimize the associated damage, we focus on the security precautions. One way turns to the intrusion detection system (IDS). An

IDS is an important security barrier that can quickly detect intrusion and security risks in the network [9, 10]. The detection algorithm is one of the most important parts in IDS. An intrusion detection algorithm suitable for critical infrastructures can accurately and quickly detect intrusions [11]. Therefore, to adapt to the new paradigm of FC, the present study focuses on the intrusion detection scheme to ensure the security and meet new challenges.

This paper will first analyze the security problems in FC/MEC. According to fog nodes/MEC hosts' resource-constrained characteristics, an intrusion detection scheme with a lightweight algorithm is proposed. This design takes full advantage of the computing resources of fog nodes/MEC hosts. It deploys classifiers for intrusion detection on fog nodes/MEC hosts and stores training data sets in the cloud server. The contribution of this work is as follows:

(1) According to the network characteristics of fog, this paper proposes a general scheme of intrusion detection system in FC/MEC environment.

(2) Based on the traditional ELM, this paper innovatively adds the sample selection process in the training phase. This design is used to improve the classifier algorithm so that it can become more lightweight when fog nodes/MEC hosts perform tasks.

(3) We compare the performance of BP, SVM, ELM, and SS-ELM as intrusion detection classifiers and verify the superiority of SS-ELM.

The paper is organized as follows. In Section 2, we will review related works on FC/MEC security and extreme learning machine- (ELM-) based IDSs. In Section 3, we will discuss the requirements and schemes of an IDS in an FC/MEC environment. In Section 4, we will introduce an ELM-based intrusion detection algorithm for FC/MEC. Section 5 will describe the simulation to verify the algorithm. The paper is concluded in Section 6.

2. Related Works

Existing research on FC/MEC security mainly focuses on analyzing security threats in FC/MEC and the corresponding countermeasures [7, 12–14].

Dsouza et al. [8] first described the advantages of FC as a new paradigm and elaborated the security problems in several different scenarios of FC, including intelligent instrument authentication, MIM attacks, and privacy issues in FC/MEC. Yi et al. [7] suggested that FC security is a problem worthy of studying, particularly the aspects of authentication, access control, intrusion detection, and user privacy issues. FC/MEC's possible contribution to network security has been discussed from the perspective of computer forensics [13, 14]. In particular, Wang et al. compared cloud computing and FC/MEC in terms of security [14] and noted that FC/MEC and honeypot technology can be integrated into cloud computing forensics to protect the security of cloud servers.

There are some studies related to intrusion detection in FC/MEC [5, 15, 16]. The security of edge computing has previously been reviewed [5], and the author suggested that FC/MEC is a paradigm of edge computing. In that paper, various security issues encountered in edge computing and security mechanisms were discussed in detail. In addition, the author noted that a fog IDS should meet the defense needs of both local fog nodes/MEC hosts and the entire network, be able to detect lasting threats, and have an intrusion prevention mechanism that can operate autonomously. It has been noted that an IDS can be deployed on the fog node/MEC host side (host-side detection) or fog network/MEC network side (network-side detection) to monitor host-side intrusions or network-side attacks [15]. A new cloudlet mesh security architecture was proposed based on cloudlets [16], and this architecture is used to protect the mobile cloud network. If cloudlet servers are treated as fog nodes/MEC hosts, then such architecture constitutes a host-side IDS.

FC/MEC is an extension computing paradigm of cloud computing. Therefore, we refer to the virtual machine (VM) IDS in cloud computing in the present study. For example, an intrusion detection scheme deployed in a cloud VM was introduced, which was based on the Smith-Waterman algorithm to collect and detect the anomalous behavior of VMs [17]. This algorithm improved the system's detection efficiency and thus reduced the detection time to some extent. In a previous report [18], attacking threats in federated cloud environments were analyzed, and a detection system for misuse cataloging was proposed. An approach named VAED which is deployed in VMs was given [19]. Its results seem to be promising for detecting evasion-based malware attacks.

Many studies focused on intrusion detection. However, since the scenarios of these studies are in the cloud, there is no focus on IDS in resource-constrained environments, especially FC/MEC. That is to say, lightweight IDS is worth studying in FC/MEC.

Intrusion detection techniques include statistics-based detection, data-mining-based detection, expert-system-based detection, support vector machine- (SVM-) based detection, genetic-algorithm-based detection, and neural-network-based detection. Considering the heterogeneity and dynamic characteristics, rich network resources, and complex attack behaviors in FC/MEC, we will use the ELM-based algorithm to design and implement an intrusion detection scheme in FC/MEC in this paper. The ELM is a neural network method. At the end of this section, we will review the application of an ELM in intrusion detection.

The ELM was first proposed by Huang et al. [20]. This algorithm uses a random mechanism to reduce the number of parameters to set and select and thus is a simple and fast learning algorithm. ELMs have been widely used in many fields since their proposal, including intrusion detection. An ELM was used for intrusion detection and showed greater accuracy in intrusion detection than that of an SVM [21]. Ye and Yu proposed an intrusion detection method in which each class was combined into an ensemble classifier using a one-to-all strategy [22]. A weighted ELM was also proposed for intrusion detection [23]. Cai et al. proposed a new fusion method which combined with a ball vector machine (BVM), ELM, and a back propagation (BP) neural network for intrusion detection. This method has shown strong performance in detection accuracy and false positive rate [24].

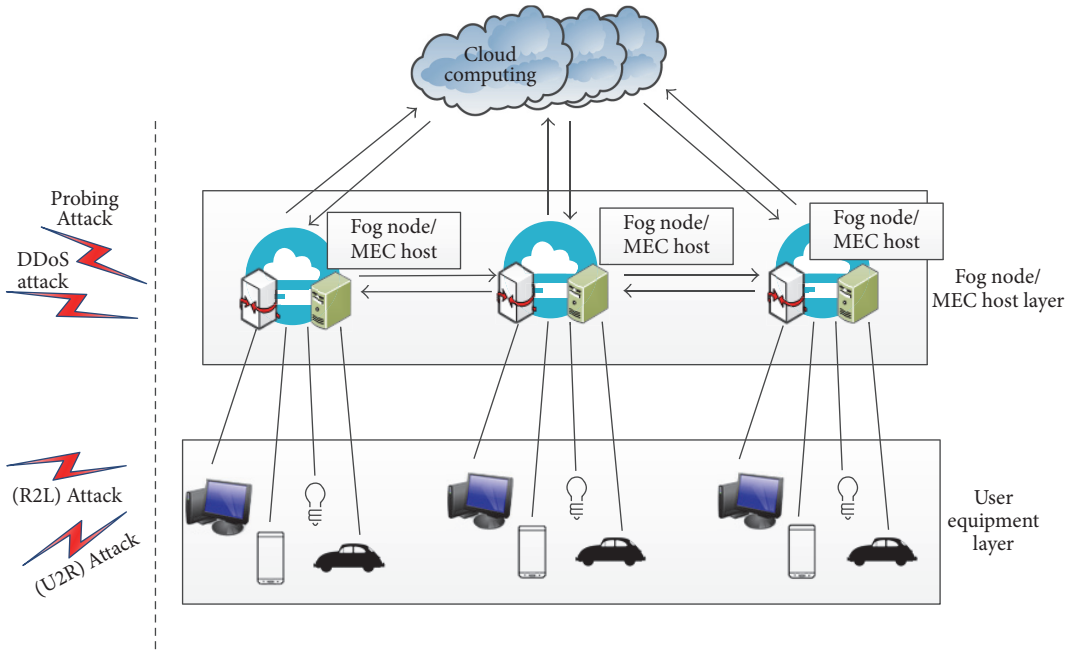

Figure 1: FC/MEC network security threats.

3. Intrusion Detection Scheme in FC/MEC

3.1. FC/MEC Network Architecture and Its Security Threats. Fog networks/MEC networks consist of user devices, fog nodes/MEC hosts, and cloud computing centers. The typical differences from the traditional cloud computing paradigm lie in the following. (1) A cloud computing center manages and controls multiple fog nodes/MEC hosts. (2) Fog nodes/MEC hosts located at the edge of the network between the network center and the user have a certain computing power. (3) Fog nodes/MEC hosts can handle computing tasks and can provide network services to user devices directly.

User devices are heterogeneous and include smart sensors, smart phones, UAVs, and other networkable terminals. We hypothesize that fog nodes/MEC hosts and terminal devices in a fog network/MEC network are all likely to be attacked and are thus not reliable. Considering the fog network/MEC network structure, we present the intrusion diagram of a network attack in FC/MEC as follows.

As shown in Figure 1, fog nodes/MEC hosts/MEC hosts in the FC/MEC layer provide a network connection service to user devices and also exchange data with the cloud computing server. Security threats come from direct or indirect attacks on fog nodes/MEC hosts/MEC hosts and user devices. In FC/MEC, network nodes are widely distributed and lack effective physical protection. They are vulnerable to invasion by malicious attackers. The detection ability and response speed of traditional IDS have been unable to meet the needs of detection of fog environment. Therefore, the establishment of an efficient intrusion detection system in FC/MEC environment has become an important research direction of FC/MEC data security.

In a fog network/MEC network, the detection of intrusion is the process in which the IDS determines whether the host state or network connection data are legal through a security audit. In addition, the fog network/MEC network IDS deployment is related to the security of the entire system. Both the fog nodes/MEC hosts/MEC hosts and the cloud computing center can compute and store data. In order to distribute the computational load of IDS reasonably, we decided to deploy IDS in the fog network/MEC network using cloud servers' storage capability and fog nodes/MEC hosts' limited computing power. There are some reasons for the deployment scheme. (1) From the perspective of detection efficiency, if an IDS is deployed in the cloud computing center only, then the network communication cost will increase because all data must be sent to and processed by the cloud server. Furthermore, the computation load of the cloud server will also increase. (2) From the perspective of security, the cloud center as the central node of the entire network will easily become the target of attacks. Once the central node is attacked, data obtained from fog node/MEC host detectors become unreliable. (3) Intrusion detection data sets are needed for training if a neural network is used for intrusion detection. Training data sets are usually large and thus will greatly increase the training time if the training is conducted only by the cloud computing center, compromising the training efficiency. (4) Due to the heterogeneity of user devices, different fog nodes/MEC hosts can result in greatly different network environments in FC/MEC. For example, certain fog nodes/MEC hosts mostly provide networking services to cars and thus communicate with vehicle sensors or car control systems, while other fog nodes/MEC hosts serve primarily smartphones. Moreover,

the customer population of a fog node/MEC host is also dynamically changing: a user device may join or exit a fog node/MEC host at any time.

3.2. Intrusion Detection Scheme in FC/MEC. As described in Section 3.1, the relationship between a fog node/MEC and user devices is of highly dynamic variability. To adapt to the dynamic fog network/MEC network and to protect the security and high efficiency of fog IDS, we propose a general architecture for FC/MEC intrusion detection systems. This scheme takes advantage of the computing capability and storage capacity of fog nodes/MEC hosts and cloud servers. The architecture includes data processing, detection, and knowledge mining in IDS. The scheme is divided into 6 layers according to the data flow of fog network/MEC network.

User Equipment Layer. FC/MEC user equipment is heterogeneous, including personal computers (PC), intelligent terminals, vehicles in Internet of Vehicles (IOV), and sensors. These devices may access different fog nodes/MEC hosts through different protocols.

Network Layer. It provides link services for different fog network/MEC network protocols. It is responsible for receiving data transmitted from the network and user equipment layer and packing and transmitting data.

Data Processing Layer. The primary role is to deal with network intrusion data from user equipment, including packet capture, data cleaning, data filtering, and data preprocessing. The task of this layer is done on fog nodes/MEC hosts.

Detection Layer. After the intrusion data is pretreated, the detection layer is sent to the classifier to detect the attacks. (1) It uses the classifier to analyze the intrusion data to determine what kind of attack it belongs to. (2) This layer is equipped with a security monitoring system to monitor the host state of fog nodes/MEC hosts. (3) At the same time, it manages and records network protocols and logs for fog network/MEC network packets. After collecting and storing a certain amount of data, the fog node/MEC host sends the test results and the relevant logs to the cloud server. In the IDS of FC/MEC, the detection layer is the core layer, and the detection task is completed in fog node/MEC host.

Analysis Layer. This layer is deployed in the cloud computing center. Its main function is to analyze the results and related logs reported by fog nodes/MEC hosts. It can integrate information into knowledge and generate some application services. For example, this layer can generate security status reports for a fog node/MEC host and store them on the cloud server.

Management Layer. Management in the cloud server is mainly responsible for (1) uniform monitoring and management of the safety status of fog nodes/MEC hosts, (2) the decision and response of the intrusion detection system, and (3) the data and logs of the intrusion fog nodes/MEC hosts that can be stored so as to facilitate intrusion forensics.

As shown in Figure 2, the intrusion detection classifier of the detection layer is the most important part of the system for intrusion data processing. It is related to the subsequent cloud server intrusion response to ensure the security of the system. Therefore, our work focuses on the detectors of intrusion attacks. Deploying detectors on fog nodes/MEC hosts can make full use of the computation capability and storage capacity of fog nodes/MEC hosts for intrusion detection. However, large-scale data cannot be processed or stored due to the limited computation and storage capability of fog nodes/MEC hosts. In addition, as described in Section 3.1, simply deploying the detector on fog nodes/MEC hosts cannot meet the requirements of a dynamic network. The cloud server has a larger storage space than fog nodes/MEC hosts; thus, the cloud server could store a large number of training sets and training set selecting rules. It can distribute training sets to fog nodes/MEC hosts dynamically for training so that different fog nodes/MEC hosts become specific and dynamic. Figure 3 shows the workflow of this scheme's steps as follows:

(1) The terminal accesses the fog node/MEC host and establishes a connection with the fog node/MEC host. The network environment of each fog node/MEC host is different.

(2) Fog node/MEC host cluster is controlled by the cloud server, so the total training set is managed by the cloud server. Store the total training set in the cloud server, and select a sample according to rules. The premise of selecting samples is that the cloud server has a perception of the network environment of fog nodes/MEC hosts.

(3) The cloud server sends the selected training set to the fog node/MEC host.

(4) Complete the training process on the fog node/MEC host.

(5) The communication between the fog node/MEC host and the terminal generates a data stream. Intrusion detection is performed on fog nodes/MEC hosts.

At the fog node/MEC host layer, intrusion detection and real-time response are required to ensure the safety and reliability of fog nodes/MEC hosts. For example, when external intruders attack a fog node/MEC host, the fog node/MEC host should be able to detect the type of intrusion and issue an alarm to prevent intrusion. Because of its computing and storage capabilities, a fog node/MEC host can complete the computation task locally to ensure short latency. We can deploy lightweight and energy-efficient algorithms on fog nodes/MEC hosts for intrusion detection.

4. SS-ELM for FC/MEC Intrusion Detection

In FC/MEC, fog nodes/MEC hosts are responsible for providing network services for massive heterogeneous intelligent devices, as the members of smart devices that provide services by a fog node/MEC host are dynamically changing. In other words, a fog node/MEC host may release a device's service or establish a connection with the device at any time. This creates a dynamic network security threat in FC/MEC. For the intrusion detection system of the fog node/MEC host, it is necessary to train the classifier with the new targeted

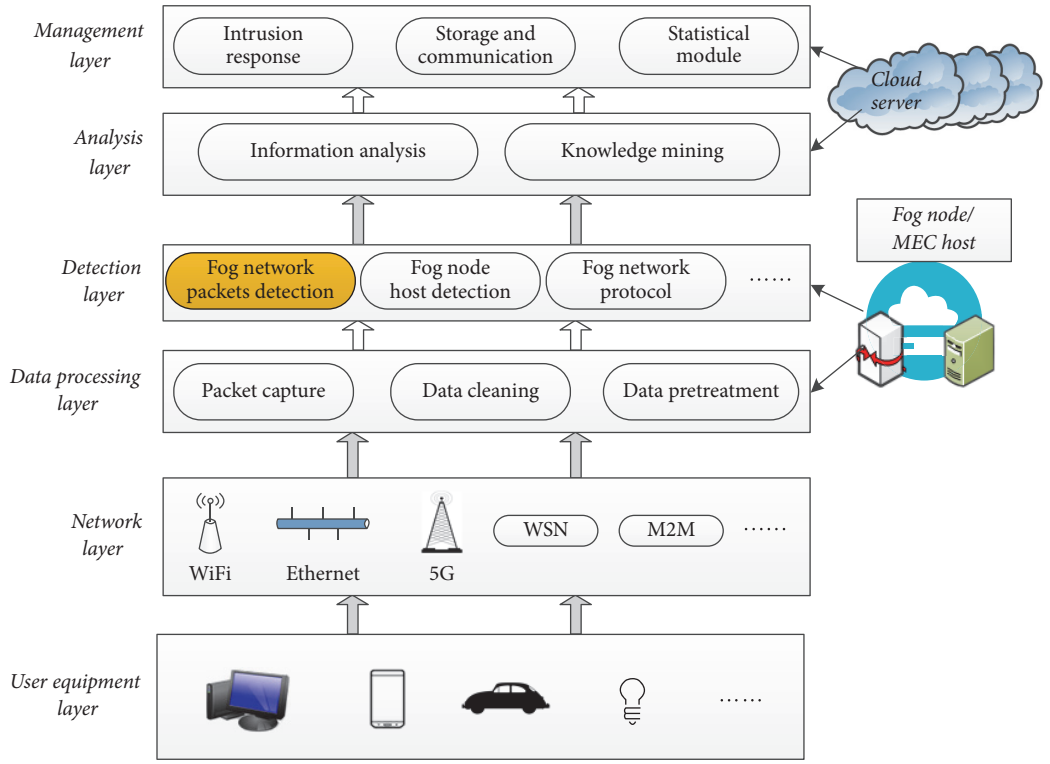

Figure 2: General architecture for FC/MEC intrusion detection systems.

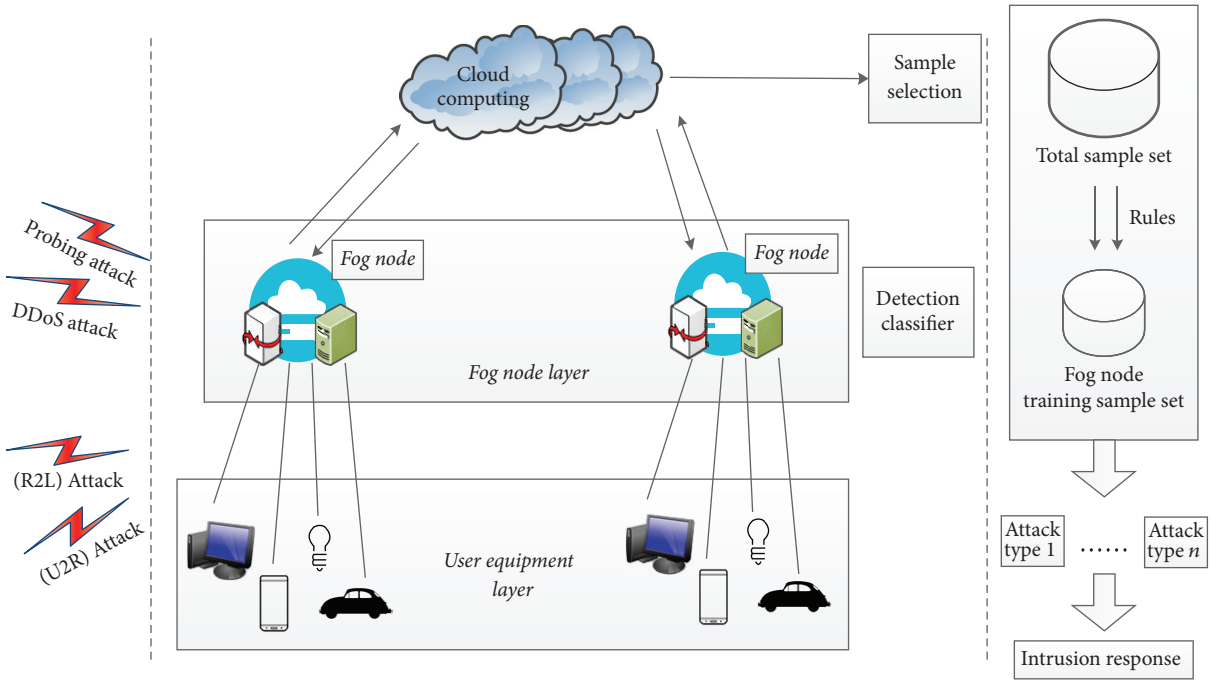

Figure 3: Fog network/MEC network intrusion detection scheme.

training set in real time. Many of the previous intrusion detection classifier algorithms have long training time [19]. Due to the limited computing power and storage capacity of fog nodes/MEC hosts, these intrusion detection schemes are not suitable for FC/MEC paradigm. Therefore, we need to study the application of lightweight intrusion detection algorithm on fog nodes/MEC hosts.

A well-suited classifier algorithm is the most important factor for a fog node/MEC host IDS. The solution for the ELM is straightforward and can be found by finding the minimum norm of a least square problem, which can ultimately be transformed into a Moore-Penrose generalized inverse problem involving a matrix. Therefore, this algorithm is characterized by a small number of training parameters, fast training speed, and satisfactory generalization ability, and thus it is suitable to be used as a classifier algorithm and deployed on fog nodes/MEC hosts. To adapt to the dynamic process of fog nodes/MEC hosts and to reduce the training time of fog nodes/MEC hosts, we select training samples according to the network characteristics and training characteristics of each fog node/MEC host. This chapter describes an algorithm for an SS-ELM for FC/MEC. We describe the principles and processes of traditional ELM algorithms in Section 4.1; furthermore, Section 4.2 describes our proposed SS-ELM algorithm. As described in Chapter Three, the algorithm is characterized by adding sample selection in the cloud server and using the deployed ELM classifier to detect network attacks on fog nodes/MEC hosts.

4.1. ELM Algorithm. ELM was first proposed by Huang of Nanyang Technology University [20]. In this subsection, the basic principles of ELM will be introduced as follows.

The ELM has a more simple and valid study mode relative to the traditional BP algorithm. Therefore, the learning speed of ELM is much faster than that of BP. In addition, traditional BP usually has some problems such as having a local minimum, being inappropriate, and having an overfitting learning rate. It, therefore, usually adopts some special methods to avoid these problems. The ELM does not need to consider these tiny problems and can get the solution of the problem directly. It is simpler than the algorithm of feedforward neural networks. Hence, ELM is more convenient and practical than the traditional ANN model. The calculation construction of ELM algorithm in this investigation is shown in Figure 4.

For the training data sample $(\vec{x}, y)$, the output function expression of single-hidden layer ahead with L hidden layer neurons to the neural network is

$$f_L(x_l) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), \tag{1}$$

where a_i and b_i are weight vector connecting i_{th} hidden neuron and the input neurons, β_i shows the output weight connecting the ith hidden layer with model output, and $G(a_i, b_i, x)$ shows the i_{th} hidden layer relative to hidden layer node output of the sample $(\vec{x}, y)$; the expression of $G(a_i, b_i, x)$ is

$$G(a_i, b_i, x) = g(a_i \cdot x + b_i). \tag{2}$$

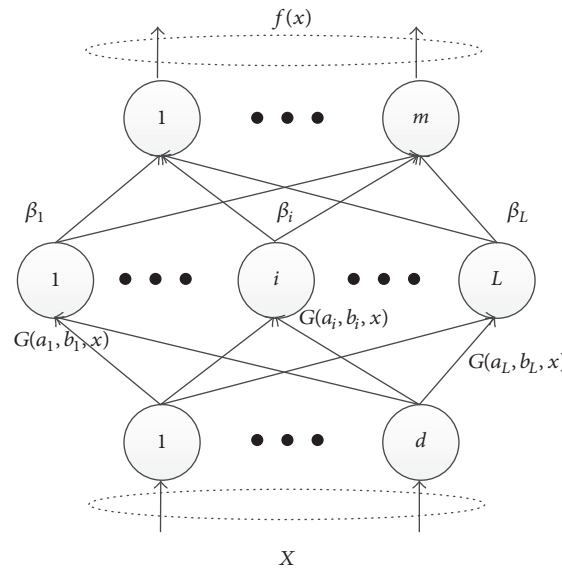

Figure 4: The architecture of the ELM neural network model in this work.

In the expression, $g : R \to R$ is the activation function and $a_i \cdot x$ is the inner product of input weight a_i and sample X in R^n. Consider N inequality data samples $\{(x_i, t_i)\}_{i=1}^{N} \subset R^n \times R^m$, if the single-hidden layer neural network of L hidden layer neurons approaches N inequality data sample with zero error; that is to say, there are a_i, b_i, and β_i, $i = 1, \ldots, L$, which makes formula (2) written as

$$f_L(x_j) = \sum_{i=1}^{L} G(a_i, b_i, x) = y_j, \quad j = 1, \ldots, N. \tag{3}$$

The shorthand formula of (3) is as follows:

$$\mathrm{H}\beta = Y, \tag{4}$$

where H is called the hidden layer output matrix; the corresponding i_{th} column shows the output of i_{th} hidden neural layer corresponding to the input $x_1, x_2, x_3, \ldots, x_n$ and the j_{th} line shows all the hidden layers relative to output amount of inputting x_j.

$$\begin{aligned} &\mathrm{H}_0(a_1, \ldots, a_L, b_1, \ldots, b_L, x_1, \ldots, x_N) \\ &= \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L}, \end{aligned} \tag{5}$$

β is output weight, $\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}$, and $Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times m}$.

In most cases, the number of hidden nodes is much smaller than the number of training samples ($L \ll N$), making it challenging for the constructed single-hidden-layer neural network (a total of L neurons in the hidden layer) to infinitely approach these mutually different samples with zero

error, resulting in errors between the network output of the training samples and the actual output. In this case, (4) can be rewritten as

$$\mathrm{H}\beta = Y + E, \tag{6}$$

in which $E = \begin{bmatrix} e_1^T \\ \vdots \\ e_N^T \end{bmatrix}_{N\times m}$. The square loss function can be defined as

$$J = \sum_{j=1}^{N} \left(\beta_i G\left(a_i, b_i, x_j\right) - y_j\right). \tag{7}$$

Equation (7) can be written as follows:

$$J = \left(\mathrm{H}\beta - \mathrm{Y}\right)^T \left(\mathrm{H}\beta - \mathrm{Y}\right). \tag{8}$$

Then, the training of the network parameters is transformed into the problem of minimizing the square loss function, that is, finding the least square solution β so that

$$\left\|\mathrm{H}\widehat{\beta} - Y\right\| = \min_{\beta} \quad \left\|\mathrm{H}\beta - Y\right\|. \tag{9}$$

The following equation can be obtained using the Moore-Penrose generalized inverse:

$$\widehat{\beta} = \arg\min_{\beta} \quad \left\|\mathrm{H}\beta - Y\right\| = \mathrm{H}^{+}Y, \tag{10}$$

in which $\mathrm{H}^{+} = (\mathrm{H}^T\mathrm{H})^{-1}\mathrm{H}^T$. If the hidden layer output matrix is not of full column rank, then the optimal external weight can be obtained using the singular value decomposition (SVD) method.

Shown above is the process of using the traditional ELM algorithm to solve the problem. In the ELM algorithm, the hidden layer node parameters are randomly determined during parameter training (in actual application, the hidden layer node parameter values are often randomly set within the interval $[-1, 1]$ because the experimental samples must be standardized).

4.2. SS-ELM Algorithm. Sample sets in the cloud server (Z_n) can be divided into two parts: sample sets to be distributed to fog nodes/MEC hosts, $Z_{nf}^f = \{z_j^f = (x_j^f, y_j^f)\}_{j=1}^{nf}$, and backup sample sets, $Z_{nc}^c = \{z_j^c = (x_j^c, y_j^c)\}_{j=1}^{nc}$, in which $Z_{nf}^f \cup Z_{nc}^c = Z_n$; $Z_{nf}^f \cap Z_{nc}^c = \varnothing$. The purpose of sample selection is to select Z_{nf}^f from Z_n, so that the network learned from Z_{nf}^f using the ELM-based algorithm can satisfy $J(a_i, b_i, \beta) \le \sigma$, in which $\sigma \in (0, \min J(a_i, b_i, \beta))$ is the predetermined upper limit of the performance index.

Assuming that $U^c = \{u_j^c \mid u_j^c = |y_j^c - f(x_j^c, a_i, b_i, \beta)|\}_{j=1}^{nc}$ is the absolute error between the sample output and network output of Z_{nc}^c, $Z_s^c = (x_m^c, y_m^c)$ is defined to include any elements in Z_{nc}^c that correspond to the maximum element in U^c. After initializing the sample sets (let $Z_{nf}^f = \varnothing$ and $Z_{nc}^c = Z_n$) and the network parameters (let $a_i = \varnothing, b_i = \varnothing, \beta = \varnothing$), the following learning rules are followed.

Rule 1.

$$Z_{nf}^f = Z_{nf}^f \cup \{z_s^c\}. \tag{11}$$

Rule 2.

$$Z_{nc}^c = Z_{nc}^c - \{z_s^c\}. \tag{12}$$

Rule 3.

$$\begin{aligned} a_i' &= a_i + \frac{\{x_s^c\} - \{x_s^c\}_{\min}}{\{x_s^c\}_{\max} - \{x_s^c\}_{\min}}. \\ b_i' &= b_i + \frac{\{x_s^c\} - \{x_s^c\}_{\min}}{\{x_s^c\}_{\max} - \{x_s^c\}_{\min}}. \end{aligned} \tag{13}$$

Rule 4. Use Z_{nf}^f, a_i, and b_i to calculate and update the optimum external weight β (let $J(a_i, b_i, \beta) = \sigma$) with (7); Z_{nf}^f, Z_{nc}^c, a_i, b_i, and β are updated. After several iterations, $J(a_i, b_i, \beta) \le \sigma$.

The procedure of the algorithm is as follows:

(1) Initialize the sample sets (let $Z_{nf}^f = \varnothing$, $Z_{nc}^c = Z_n$) and network structure parameters (let $a_i = \varnothing$, $b_i = \varnothing$, $\beta = \varnothing$) on the cloud server side.

(2) Randomly generate hidden layer node parameters on fog node/MEC host (a_i, b_i), $i = 1, \ldots, L$.

(3) Calculate the hidden layer output matrix H (ensure that H is of full column rank).

(4) Calculate $J(a_i, b_i, \beta)$ and U^c.

(5) If $J(a_i, b_i, \beta) \le \sigma$, turn to step (10); otherwise, continue.

(6) Select Z_s^c from Z_{nc}^c, $Z_{nf}^f = Z_{nf}^f \cup \{z_s^c\}$; $Z_{nc}^c = Z_{nc}^c - \{z_s^c\}$.(7) Update a_i and b_i. $a_i' = a_i + (\{x_s^c\} - \{x_s^c\}_{\min})/(\{x_s^c\}_{\max} - \{x_s^c\}_{\min})$; $b_i' = b_i + (\{x_s^c\} - \{x_s^c\}_{\min})/(\{x_s^c\}_{\max} - \{x_s^c\}_{\min})$.

(8) Use Z_{nf}^f, a_i, and b_i to calculate and update the optimum external weight β (let $J(a_i, b_i, \beta) = \sigma$) with (7).

(9) If $Z_{nc}^c \neq \varnothing$, execute step (4); otherwise, continue.

(10) Use Z_n, a_i, and b_i to calculate and update the optimum external weight β with (7), and the algorithm ends here.

In this work, the hidden layer activation function of ELM model adopts a sigmoid transformation function:

$$G\left(a_i, b_i, x\right) = \frac{1}{1 + e^{-(a_i x + b_i)}}. \tag{14}$$

An algorithm analysis shows that most of the time taken for selecting samples for fog nodes/MEC hosts was spent in calculating $J(a_i, b_i, \beta)$. Assuming that $t(J)$ is the average time for calculating $J(a_i, b_i, \beta)$ and that the delay caused by data transmission between the cloud computing and fog nodes/MEC hosts is t' ($t' \ll t(J)$), then the learning time for fog nodes/MEC hosts is $t(N) \approx n_l \cdot t(J)$.

5. Numerical Simulation

At present, there are no training sets available for FC/MEC intrusion detection. Therefore, we used the KDD Cup 99

Table 1: Comparison of algorithms in testing accuracy and training time.

Algorithm	Accuracy (%)	Training time (s)
SS-ELM	99.07 ± 0.11	4.52 ± 0.10
ELM	96.09 ± 0.07	4.15 ± 0.04
BP	84.16 ± 0.18	387.92 ± 0.21
SVM	94.25 ± 0.08	15.02 ± 0.14
CVM-ELM	96.87 ± 0.23	12.47 ± 0.53

data set [25] for the analysis in the simulation experiments. In the data set, 41 fields were collected for each network connection. The abnormal types were divided into four major categories of 39 attack types, of which 22 attack types were presented in the training set and 17 unknown attack types were presented in the test set. In addition, the KDD99 connection records contained both symbolic features and discrete features. Therefore, the symbolic features needed to be converted prior to the experiment. We used the data preprocessing scheme reported in the literature [26]. The data preprocessing primarily includes converting category features into metric features and then normalizing metric features to prevent greater metric features from dominating and outweighing smaller features. For the supervised learning and prediction conducted in the experiment, all features were normalized to be within $[0, 1]$. The cloud server was simulated in Windows 7 operating system (i7-2760QM, 2.4 GHz CPU, 8.00 GB RAM), and the SS-ELM algorithm was implemented using Matlab 2014a.

We compared the performance of various fog IDS classifiers, including the SS-ELM, traditional ELM, BP neural network, SVM, and CVM-ELM [27]. The ELM, BP, and SVM were deployed on fog nodes/MEC hosts. The kernel function of SVM was *rbf*, $gamma = 0.005$, and $C = 10$. The BP learning rate was 0.06, the momentum coefficient was 0.9, the maximum number of epochs (iterations) was 5,000, and the goal was 0.0001. A total of 2,000 and 10,000 pieces of data were selected from the KDD Cup 99 and used as training samples and test samples, respectively. The training time and detection accuracy were compared. The experimental data were the average of 10 runs.

According to the results in Table 1, the SS-ELM has the highest accuracy, which proves that SS-ELM is the most suitable classifier algorithm for fog node/MEC host deployment, while from the training time perspective, the SS-ELM requires a slightly longer training time than the traditional ELM because sample selection is required in the SS-ELM. BP performed poorer than the SS-ELM and ELM in terms of training time and accuracy, and the SVM and CVM-ELM also require a longer training time. Therefore, in terms of accuracy, SS-ELM is more suitable for intrusion detection in FC/MEC.

In experiment 2, we analyzed the training time and accuracy of the SS-ELM algorithm for training sets of different scales. Based on the results of experiment 1, we selected only SS-ELM, ELM, CVM-ELM, and SVM for comparison

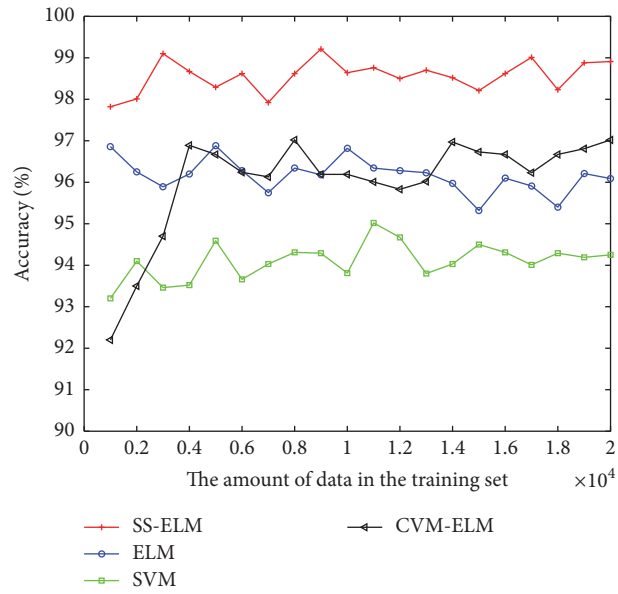

Figure 5: Accuracy rate comparison of SS-ELM, ELM, SVM, and CVM-ELM.

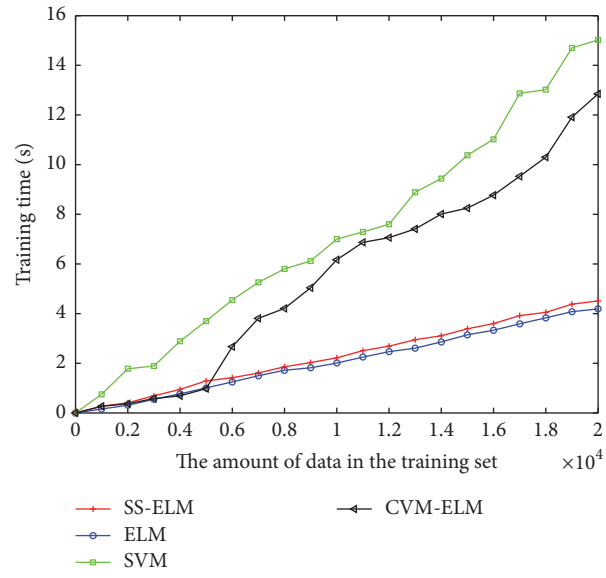

Figure 6: Training time comparison of SS-ELM, ELM, and SVM.

of the training time and accuracy. The training set sizes were selected as 1,000, 2,000,. . ., 20,000. The experimental results are shown in Figures 5 and 6.

As shown in Figure 5, as the size of the training set increases, SS-ELM shows the highest intrusion detection accuracy in FC/MEC relative to that of ELM and SVM. We analyze it from the aspect of algorithm, and we think that the main reason is the following:

(1) Unlike the traditional classic gradient-based learning algorithms which intend to reach minimum training error but do not consider the magnitude of weights, the ELM tends

to reach not only the smallest training error but also the smallest norm of weights.

(2) Unlike the traditional classic gradient-based learning algorithms facing several issues like local minimum, improper learning rate, overfitting, and so forth, the ELM tends to reach the solutions straightforwardly without such trivial issues.

(3) In our improved algorithm, we optimize the square loss function $J = \sum_{j=1}^{N}(\beta_i G(a_i, b_i, x_j) - y_j)$ for each fog node/MEC host in the training phase. This optimization is mainly reflected in the process of sample selection. According to formula (7) to formula (10) in the manuscript, each fog node/MEC host has an optimal external weight β in the network environment which is more suitable for the training.

For the training time, as shown in Figure 6, the simulation result shows that the training time of our proposed SS-ELM is slightly less than of ELM. The main reason is that SS-ELM has more than one sample selection step compared with ELM. Although the SS-ELM performed worse than the ELM, the difference is small and within an acceptable range. Therefore, we conclude that SS-ELM algorithm performs better in intrusion detection of FC/MEC.

As noted in Section 3, fog nodes/MEC hosts in the FC/MEC network are highly dynamic. Thus, we must analyze the robustness of the algorithm designed; specifically, we must analyze the dependency of SS-ELM algorithm on time statistics. In the KDD99 data set, there are 9 features (features 23–31) that are about time-based traffic features of the connection records. These features are based on time statistics. They can present some relationships between the current connection records and the connection records in the previous period. However, in the real FC/MEC network environment, there are a large number of raw data without manual statistical processing. That is to say, it is very difficult for us to obtain data based on time statistics. In order to ensure that the classifier algorithm can work effectively in the network environment with high real time and high dynamics, we removed features 23–31 to carry on experiment 3. In addition, new data sets obtained were used to compare the four algorithms used in experiment 1. The results are shown in Figure 7.

Figure 7 shows that the accuracy of SS-ELM did not decrease significantly after removing the time statistical features, while the ELM, BP, SVM, and CVM-ELM showed that accuracy decreases to various extents, with the decrease of the BP algorithm being the greatest. From the data of experiment 3, we can conclude that the SS-ELM has the least dependency on time attribute and is suitable for network environments that are highly dynamic and feature large changes.

In experiment 4, to analyze the false positive rate of the SS-ELM, we showed the performance of SVM, SS-ELM, and ELM in terms of the receiver operating characteristics (ROCs) [28]: for the ROC curve, the x-axis represents the false positive rate, and the y-axis represents the true positive rate. A classifier algorithm with a larger area under the ROC curve (AUC) performs better. Figure 8 shows that SS-ELM outperformed the other two algorithms in FC/MEC.

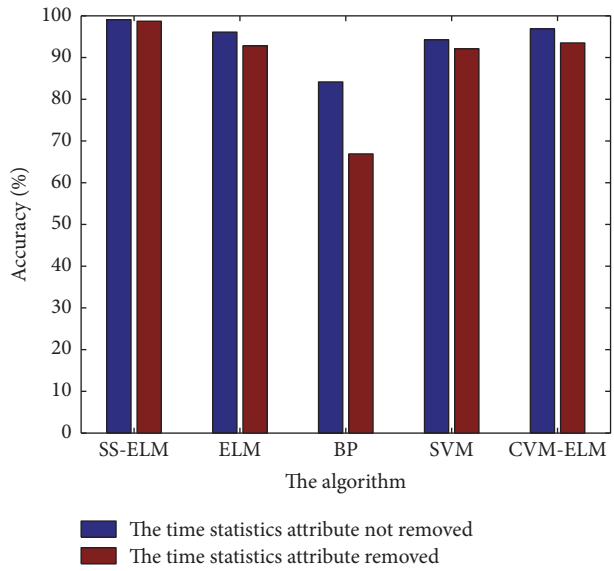

Figure 7: A comparative study on the dependence of time-dependent attributes.

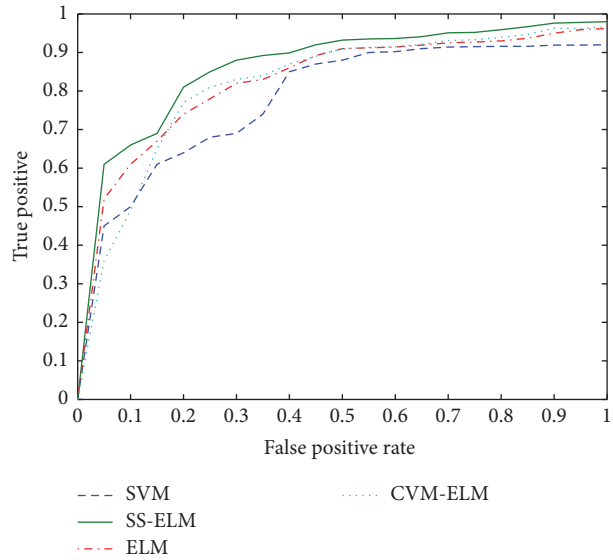

Figure 8: ROC of SVM, SS-ELM, ELM, and CVM-ELM.

6. Conclusions

FC/MEC is a new paradigm. Fog nodes/MEC hosts' resource-constrained features bring about new problems to the security realm, particularly in the area of intrusion detection. For MEC, its characteristics are similar to FC/MEC. In the present study, an FC/MEC IDS scheme was established based on the analysis of the network characteristics and security requirements of FC/MEC. In addition, an SS-ELM algorithm that combines sample selection and ELM is proposed according to the network characteristics of FC/MEC. The training time and detection accuracy of ELM, BP, and SVM are examined experimentally. In the experiment, the SS-ELM

algorithm shows excellent performance in FC/MEC, particularly in accuracy and time dependence. Several experiments have demonstrated the advantages of SS-ELM as a lightweight algorithm from different perspectives and have contributed to solving the problems caused by resource constraints in FC/MEC.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Science and Technology Key Projects (no. 61602034) and the U.S. National Science Foundation under Grants CNS-1559696 and IIA-1301726.

References

[1] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[2] E. Portal, "Mobile-edge computing-introductory technical white paper," Tech. Rep., 2014.

[3] M. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proceedings of the the 6th International Conference on Advances in Future Internet*, 2014.

[4] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, In press.

[5] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[6] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications Magazine*, vol. 23, no. 5, pp. 120–128, 2016.

[7] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the Workshop on Mobile Big Data (Mobidata '15)*, pp. 37–42, ACM, Hangzhou, China, June 2015.

[8] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: preliminary framework and a case study," in *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IEEE IRI '14)*, pp. 16–23, Redwood City, Calif, USA, August 2014.

[9] D. E. Denning, "An Intrusion-Detection Model," in *Proceedings of the IEEE Symposium on Security and Privacy*, p. 118, Oakland, Calif, USA, April 1986.

[10] S. Wang, Q. Sun, H. Zou, and F. Yang, "Detecting SYN flooding attacks based on traffic prediction," *Security Communication*, vol. 5, pp. 1131–1140, 2012.

[11] O. Linda, T. Vollmer, and M. Manic, "Neural Network based intrusion detection system for critical infrastructures," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2009*, pp. 1827–1834, Atlanta, Ga, USA, June 2009.

[12] I. Stojmenovic and S. Wen, "The fog computing paradigm: scenarios and security issues," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '14)*, pp. 1–8, IEEE, Warsaw, Poland, September 2014.

[13] S. J. Stolfo, M. B. Salem, and A. D. Keromytis, "Fog computing: Mitigating insider data theft attacks in the cloud," in *Proceedings of the 1st IEEE Security and Privacy Workshops, SPW 2012*, pp. 125–128, San Francisco, Calif, USA, May 2012.

[14] Y. Wang, T. Uehara, and R. Sasaki, "Fog computing: Issues and challenges in security and forensics," in *Proceedings of the 39th IEEE Annual Computer Software and Applications Conference Workshops, COMPSACW 2015*, pp. 53–59, Taichung, Taiwan, July 2015.

[15] S. Yi, Z. Qin, and Q. Li, "Security and Privacy Issues of Fog Computing: A Survey," in *Wireless Algorithms, Systems, and Applications*, vol. 9204 of *Lecture Notes in Computer Science*, pp. 685–695, Springer International Publishing, Cham, 2015.

[16] Y. Shi, S. Abhilash, and K. Hwang, "Cloudlet mesh for securing mobile clouds from intrusions and network attacks," in *Proceedings of the 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015*, pp. 109–118, San Francisco, Calif, USA, April 2015.

[17] N. Pitropakis, C. Lambrinoudakis, and D. Geneiatakis, "Till all are one: Towards a unified cloud IDS," in *Trust, Privacy and Security in Digital Business*, vol. 9264, pp. 136–149, Springer International Publishing, 2015.

[18] E. C. Oscar, E. B. Fernandez, and M. A. Raúl, "Threat analysis and misuse patterns of federated Inter-Cloud systems," in *Proceedings of the 19th European Conference on Pattern Languages of Programs, EuroPLoP 2014*, Irsee, Germany, July 2014.

[19] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "VAED: VMI-assisted evasion detection approach for infrastructure as a service cloud," *Concurrency Computation Practice Experience*, vol. 29, no. Issue, p. 30, 2017.

[20] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[21] C. Cheng, W. P. Tay, and G.-B. Huang, "Extreme learning machines for intrusion detection," in *Proceedings of the Annual International Joint Conference on Neural Networks, IJCNN 2012*, Brisbane, Australia, June 2012.

[22] Z. Ye and Y. Yu, "Network intrusion classification based on extreme learning machine," in *Proceedings of the IEEE International Conference on Information and Automation, ICIA 2015*, pp. 1642–1647, Lijiang, China, August 2015.

[23] W. Srimuang and S. Intarasothonchun, "Classification model of network intrusion using Weighted Extreme Learning Machine," in *Proceedings of the 12th International Joint Conference on Computer Science and Software Engineering, JCSSE 2015*, pp. 190–194, Songkhla, Thailand, July 2015.

[24] C. Cai, H. Pan, and G. Cheng, "Fusion of BVM and ELM for anomaly detection in computer networks," in *Proceedings of the International Conference on Computer Science and Service System, CSSS 2012*, pp. 1957–1960, Nanjing, China, August 2012.

[25] KDD CUP 99 data set, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[26] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: a review," *Computers & Security*, vol. 30, no. 6-7, pp. 353–375, 2011.

[27] Y. Zhou, *An incremental ELM algorithm based on instance selection [M.S. thesis]*, Hebei University, Hebei, China, 2015.

[28] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.