Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2020.010102
ISSN 2790-0932 Vol. 1, Issue 1: 18-34

Data Center Resource Allocation Strategy Based on Edge Computing

Qiang Long*

Heilongjiang University, Harbin, China

**corresponding author*

Keywords: Edge Computing, Minimum Access Delay, Network Reliability, Resource Allocation

Abstract: In recent years, Mobile Edge Computing (MEC) technology can support resource-intensive applications in edge networks. With the expansion of cloud computing service models in edge networks, it provides users with real-time services, which solves traditional cloud computing the high latency barrier when the center provides services is a new technology with very broad application prospects. This article aims to study the data center resource allocation strategy of edge computing. This paper designs an edge server experiment to make full use of the edge computing model to mine the computing power of the edge terminal in the network, perform some or all calculations at the edge terminal, process private data, and reduce the computing, transmission bandwidth load and energy consumption of the cloud computing center. The concept of edge computing was proposed only after the development of cloud computing technology for a period of time. At present, edge computing and cloud computing technology are complementary to each other. As the technical basis for building an interconnected environment of all things, this paper studies in detail the use of mobile edge computing technology to provide users with real-time application services The architecture of this service model analyzes the challenges faced by the service model and proposes an effective solution, which improves the efficiency of resource allocation by nearly 45.73%.

1. Introduction

The rapid development of mobile wireless communication technology has brought us into a whole new world, and the era of the Internet of Everything is coming soon. The number of various mobile access devices was expected to reach 50 billion in 2020. At the same time, a large number of multimedia services are rapidly emerging, which directly leads to the rapid growth of data access to the network. According to a recent IBM report, the amount of data generated in 2020 will be 40

Copyright: © 2020 by the authors. This is an Open Access article distributed under the Creative Commons Attribution License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (https://creativecommons.org/licenses/by/4.0/).

trillion gigabytes, 44 times that of 2009. Many new mobile services, such as speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality, online planning and decision-making, etc. usually require abundant computing resources and are very sensitive to delays. The huge challenge to traditional wireless communication networks is how to provide services for our daily lives in real time through various portable mobile devices for these computationally intensive applications.

Zhang Q believes that the emergence of edge computing technology shares part of the data processing in the cloud, but with the expansion of scale, a large number of IoT terminal data processing puts forward higher requirements on the communication performance of edge computing. Some existing communication processing methods improve overall performance by reducing the amount of data, and the underlying communication efficiency problem has not been effectively solved. In response to the above problems, he proposed a DPDK-based multi-port parallel communication mechanism that can be used in edge computing node communication, and added an improved port distribution strategy and feedback control strategy. Experiments show that this mechanism not only realizes the multi-port parallel communication based on the data link layer, but also realizes the high-speed data interaction between the DPDK underlying communication process and multiple application processes, as well as the reasonable allocation of bandwidth resources according to the underlying communication control module, but he did not explain the relationship between these two needs to be further studied [1]. Aiming proposed a data collaborative caching strategy based on game theory to solve the problem of limited server storage capacity in edge computing.This strategy divides the edge computing environment into multiple regions based on the coverage of the base station, and each region cooperates with neighboring regions to cache data resources. In each area, the cache value of each data block to the local area and adjacent areas is calculated, and the cache decision is made based on the cache value of the resource to be cached to minimize the delay for users to obtain data resources. The simulation experiment results show that the proposed caching strategy reduces the average acquisition delay of data resources by 36.55% compared with the existing non-cooperative caching strategy, which effectively reduces the average acquisition delay of data resources, but he failed to take other factors into consideration., With one-sidedness [2]. Yang L believes that edge data centers have higher requirements in terms of customized service capabilities because they are closer to data sources. In response to this demand, he proposed a method for constructing RO of physical resources in edge data centers. Through the RSD architecture server, physical resource reconstruction can be realized, and various resources such as computing, storage, and network can be changed from a tightly coupled relationship to a loosely coupled relationship that can be scheduled by software, and a hardware infrastructure that matches the business load can be realized. Experimental results show that the RSD-based edge data center construction method can achieve hardware resource pooling, improve resource utilization, and is suitable for use in the edge data center environment, but this method has limitations and cannot be applied to other cases [3].

When the processing power and resources required by the locally processed application services far exceed the services that the hardware of the mobile device can provide, based on the inherent characteristics of the mobile device, the lack of resources causes many mobile applications to provide services through portable mobile devices. Users provide services. The main obstacle to providing real-time services, there is an urgent need for a solution that can replace portable mobile devices and provide high-quality mobile services to meet the highly dynamic needs of mobile users. We can solve the problem of poor mobile device resources by deploying edge servers with relatively abundant resources near mobile devices, rather than remote cloud computing centers.

Mobile edge computing technology provides mobile users with a wealth of application services by migrating service resources from the remote cloud computing center to the edge network, which can greatly eliminate the network during data transmission when the remote cloud computing center provides services. Time delay. Therefore, while bringing abundant computing capabilities to resource-poor mobile devices, mobile edge computing can provide real-time interactive responses to services by using the advantages of computing resources that are closer to mobile devices geographically, and then become a service for those resource needs. New mobile applications with intensive and high latency requirements provide more suitable solutions for services, and actually increase the reasonable and efficient resources by nearly 45.73%.

2. Data Center Resource Scheduling Related Methods

2.1. Data Center Energy Consumption Assessment Method

The energy consumption evaluation standard of the data center directly affects the accuracy and credibility of energy consumption perception related experiments. Before conducting resource scheduling algorithm optimization experiments, a suitable data center energy consumption evaluation method should be established. The existing cloud computing data center energy consumption assessment methods can be roughly divided into three categories: one is to directly measure the energy consumption of the data center, the other is the energy consumption estimation method for real physical clusters, and the third is the built-in cloud computing simulation platform. Energy consumption assessment method [4-5].

(1) Energy consumption evaluation method based on direct measurement: The direct measurement method includes the use of external instruments for measurement and the use of energy collection system for measurement. Both hardware and software-based energy consumption measurement methods can achieve more accurate energy consumption measurement [6]. In addition to using physical instruments to measure the energy consumption of the data center, some server manufacturers provide energy consumption data collection software, which runs directly on the physical cluster and can measure the energy consumption of the server in real time. Direct energy consumption measurement of data centers is the simplest and most direct energy consumption evaluation method, and has the highest accuracy. However, this evaluation method is more suitable for a single node or a small-scale cloud computing data center. The deployment of cloud computing data center is very difficult and financially expensive, and the feasibility is low [7-8].

(2) Cloud computing simulation platform built-in energy consumption evaluation method: the design of data center resource scheduling algorithm often requires a large number of experiments. If the experiment is directly performed on the physical cluster, it will need to bear huge experimental expenses, and once the intermediate results or parameters of the experiment are wrong It is necessary to restart or reconfigure the server, which leads to an excessively long experiment period. Therefore, the simulation platform is applied to the research of data center resource scheduling algorithm to make related experiments more economical and efficient. The energy consumption evaluation method applied in the simulation environment should improve the accuracy of its evaluation as much as possible on the basis of ensuring easy modeling, expansion and implementation [9-10]. The first energy consumption evaluation method directly records the total number of VMs running in the simulated data center. Since the number of virtual machines reflects the energy consumption, the energy consumption is directly evaluated by comparing the total number of virtual machines running. This method is simple and intuitive, but the accuracy is poor. The more commonly used is the energy consumption evaluation model that depends on the CPU

usage rate [11-12]. Eleven sampling data points are selected, and the sampling gradient of CPU utilization is 10%. Finally, a one-element segmented model is used to establish an energy consumption model.

2.2. Resource Scheduling Algorithm Based on Task Allocation

Task allocation in cloud computing is an important part of cloud computing resource scheduling, also known as task scheduling. The essence of cloud computing task allocation is to allocate massive application tasks submitted by users to virtual machines in the data center according to a certain strategy. What is achieved in this process is the mapping of tasks to virtual machines [13-14]. The tasks of the cloud computing data center can be summarized into two types: independent tasks and associated task scheduling. The difference between the two is whether there is a mutual dependency between tasks. The cloud computing task allocation problem is essentially an NP-hard problem. Many factors need to be considered in the task allocation process, and may not be universally applicable. There are different optimization goals for different business needs. The main optimization goals of cloud computing tasks Covering several aspects such as energy consumption, execution time, load balancing, etc., the following brief introductions are given in turn:

(1) Task allocation with the goal of reducing energy consumption: The expansion of the scale of the data center has led to an increase in the energy consumption of the cloud computing system, and energy consumption has become an inevitable consideration in the task scheduling process. The main way to reduce energy consumption is to minimize the number of computing nodes used and time occupied, shorten task execution time, and improve resource utilization [15-16]. However, energy-saving priority task scheduling is likely to have uneven load or affect system performance. Therefore, task allocation with the goal of reducing energy consumption should be carried out under the premise of ensuring system performance requirements and load balancing.

(2) Task allocation with the goal of shortening execution time: Minimizing task execution time is the most common optimization goal in cloud computing task scheduling optimization. Shortening task execution time is conducive to improving resource utilization, reducing operating costs, and improving user satisfaction, it is also conducive to the optimization of energy consumption [17].

(3) Task allocation with the goal of ensuring load balancing: Load balancing in the task allocation process of cloud computing data centers refers to the uniform allocation of service requests to multiple computing resources. Due to the large scale of the data center, which contains a large number of physical nodes and virtual machines, load balancing is also very important in the task distribution process. If load balancing cannot be achieved in the task scheduling process, resources will not be fully utilized, the performance of the system will be reduced, and energy consumption will also increase [18].

3. Edge Server Deployment Design Experiment

3.1. Definition of the Minimum Access Delay Problem of Edge Servers

All service requests arriving at AP i come from two aspects: service requests from mobile devices within the AP i data receiving range; service requests from other mobile devices within the AP data receiving range need to be routed through AP i to provide services for them Edge server [19]. Therefore, the total rate of arrival of service requests on AP i can be described by the following formula:

$$\lambda_{\mathrm{i}}=_{n,m\in V}\sum\sum i_{m,j} \tag{1}$$

The process of service request data waiting to be transmitted on the AP is just like the M/M/1 queue. Variables are used to indicate the data transmission rate of AP i. In the model proposed in this chapter, the data transmission rate is determined by the data transmission capacity of AP i and the size of the service request data volume is determined. Assuming that all APs will receive all sent data packets, based on this assumption, it can be used to evaluate the number of service requests waiting to be transmitted in the service queue in AP i [20-21].

$$min\min\frac{\sum i\in V\lambda_i/(\mu_i-\lambda_i)}{\sum_{i_{i\in V}}R_i} \tag{2}$$

From the perspective of the entire edge server access system, all service request data sent by mobile devices enter the edge server access system through the adjacent AP, and leave the system when the service request reaches the edge server that provides the service. This chapter defines the average access delay of the edge server as the average residence time of service requests in the edge server access system, and then calculates this average access delay according to Little's law [22].

$$\sum I_{\mathrm{i},j}=1\in V \tag{3}$$

In a given edge network based on SDN technology, this chapter defines how to optimize the deployment of k edge servers to minimize the average access delay for service requests sent by mobile devices in the entire network to reach the edge servers that provide services for them. The optimized deployment of K edge servers in the edge network is specifically defined as follows:

$$\sum_{\mathrm{i}\in V}I_{i,j}=\forall j+k\in C \tag{4}$$

According to the above description, the optimal deployment of edge servers in the edge network supported by SDN technology can be easily transformed into a typical k-median problem [23]. The k-median problem is not only a typical NP problem, but the optimal deployment of k edge servers in an edge network is also an NP problem.

$$\mu_{\mathrm{i}}>\lambda_i=\forall i+k\in V \tag{5}$$

3.2. Optimal Edge Server Deployment Algorithm Based on Enumeration

The degree centrality of an AP is defined as the number of APs directly connected to it, and RNOESPA uses the largest possible degree |V|-1 in the AP to normalize the degree centrality of each AP. The degree centrality of AP i is expressed, deg(i) represents the number of other APs connected to AP i in one hop [24]. Intermediate centrality is a commonly used indicator for centrality measurement. It refers to the ratio of the shortest path passing through AP i and connecting other two APs to the total number of shortest paths between these two APs. It is defined as follows:

$$\beta_{\mathrm{i}}^{b}=\sum_{s,t\in V}\sum_{j\in C}\forall i\in V \tag{6}$$

The load centrality of AP i refers to the proportion of paths passing through AP i among all routing paths in the edge network, which is defined as follows:

$$\beta_i^{\mathrm{l}} = \sum s,t \in V \sum \forall i \in V \tag{7}$$

These four connection attributes of each AP are used as indicators to measure its deployment location as an edge server. RNOESPA uses extreme value normalization to normalize the value of each connection attribute of all APs. Based on the centrality of each AP, RNOESPA uses information entropy to measure the discreteness of the value of each connection attribute on all APs [25].

$$H_\beta = -\sum_{\ln|V|} i \in V \beta_i \ln \beta_i \tag{8}$$

When the connection attribute has a higher entropy value, it means that the connection attribute of all APs can provide less information [26]. The difference between the entropy value and 1 and the entropy value of each connection attribute determine the information utilization rate of each connection attribute. The entropy weight of each connection attribute is defined as:

$$\omega_\beta = |A| - \sum^{1-H_\beta} \beta \in A H_\beta \tag{9}$$

The entropy weight of each connection attribute is used to measure the overall evaluation of each AP as an edge server and the appropriateness of this connection attribute.

$$T_v^s = \mu - \sum \in m A^m \tag{10}$$

In order to avoid that the edge servers are deployed in the same local area in the edge network, RNOESPA also considers the influence factors of the suitability of the deployed edge servers on the suitability of other candidate APs used to deploy the next edge server in the network [27-28], denoted by F. Based on these normalized indicators, RNOESPA calculates the sum of the weights of all connection attributes on each AP. The normalized impact factor of the deployed edge server is also used as a dependent variable. The appropriateness of each AP as an edge server placement location is calculated as follows:

$$_{\beta \in A} W_i = F_i \cdot \lambda_i \cdot \sum \omega_\beta \beta_i \tag{11}$$

3.3. Approximately Optimal Edge Server Deployment Algorithm

The different configurations of VRC supporting different application services in the edge network will not affect the one-hop transmission delay between the mobile device and the nearby edge server. Therefore, when evaluating the optimal configuration of VRC supporting various application services in the edge network, this chapter does not take into account the one-hop wireless transmission delay between the mobile device and the nearby edge server [29].

$$\sum_{v \in V} \mathrm{I}_v^m = k, \forall m \in V \tag{12}$$

When the edge server u requests to provide application services for mobile devices within the coverage of the edge server v, the total network delay caused is:

$$0 \le \mathrm{a}_{\mathrm{v}}^{\mathrm{m}} \le 1, \forall n_v \in U \tag{13}$$

The process of each edge server providing services for application service requests can be

regarded as an M/M/1 queuing model. Therefore, it can be deduced that the average processing delay of all service requests in the edge server v is:

$$\mu_v - \sum_M A_v^m > 0 = \forall v \in V \tag{14}$$

The limitation is to ensure that the service rate is greater than the requested arrival rate, so as to ensure the stability of the system. The restriction guarantees that the sum of the number of mobile devices covered by each edge server is equal to N, thus ensuring that no requests sent by mobile devices are double-counted [30].

$$\sum_{u \in V} n_u = N, \forall n_u \in U \tag{15}$$

4. Minimum Access Delay of Mobile Edge Servers

4.1. Edge Network Provides Multiple Application Service Architectures

According to the mobile edge network architecture based on SDN technology, this article uses Python programs to design an experimental model to simulate the process of service requests accessing the edge server. This model can well reflect the transmission of service requests when accessing the edge server in actual scenarios. Delay, and then can be used to evaluate the performance of the algorithm designed in this paper. In order to show the characteristics of the actual edge network topology, according to the random network generation method introduced above, this paper sets ϵ=2 to generate the edge network simulation topology. Due to the high cost of edge servers, the number of deployments is much smaller than the number of SDN-based access points. Therefore, this article reflects the quantitative relationship between the two in the actual scenario by setting the value range of the edge server to [1,4] and the number of APs to [10,40], as shown in Table 1.

Table 1. Table of experimental simulation parameters

Experiment parameter	Value
Number of edge servers	[1,4]
AP Amount	[10,40]
Number of mobile devices	[200,1000]
The average amount of data requested by the service	[20,100]KB
AP Failure probability	[0.05,0.08]
Link failure probability	[0.02,0.08]
Network attachment rate	2
Maximum data transfer rate	1.0 Gbps

In this paper, the transmission data rate of each AP is set to 1.0 Gbps. Considering the scale of the simulation experiment and the service request in the actual application scenario, this article assumes that no more than 1000 mobile devices in the simulation scenario send various service requests to the edge server randomly at the same time.

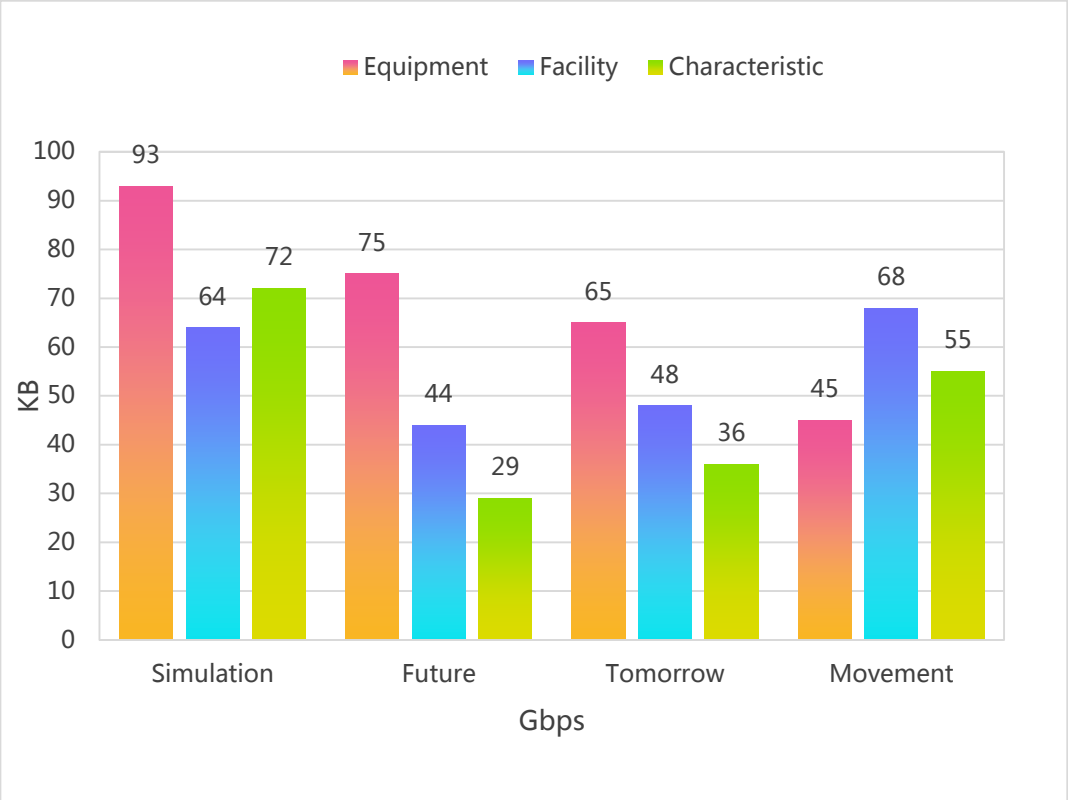


Figure 1. Features that send new application service requests

As shown in Figure 1, in order to simulate the characteristics of new application service requests sent by mobile devices in the future, this paper sets the average service request size within 100 KB based on the characteristics of mobile applications tested in several laboratories. Although the request data size is different for different application service requests, the size of the data packet in the actual transmission protocol is fixed. Therefore, in the simulation data transmission, all application service request data is divided into different numbers and sizes. Unified data packet unit.

This chapter uses a Python program to develop a simulation environment to simulate the impact of different configurations of VRC supporting multiple application services in the edge network on the service request response delay of each application. In order to reflect the performance of the algorithm designed in this chapter in actual scenarios, the network topology in an online real-world network topology database is used to represent the edge network topology in the simulation experiment, as shown in Table 2.

Table 2. Real-world network topology database

Algorithm	Computation complexity
OESDA	$O(\|M\|\cdot(\|V\|+\|M\|))$
LAHSDA	$O(\|M\|\cdot\|V\|\cdot k\cdot(\|V\|+\|M\|))$
CEHSDA	$O(k\cdot\|V\|\cdot\|M\|^2\cdot(\|V\|+\|M\|))$
SEHSDA	$O(k\cdot\|M\|^2\cdot\|V\|^3)$
GSDA	$O(k\cdot\|M\|\cdot\|V\|^2)$

First, an actual network topology with 19 nodes and 25 edges is used to verify the performance changes of the five algorithms of OESDA, GSDA, LAHSDA, CEHSDA and SEHSDA as the number of VRCs supporting each application service increases. The comparison result of the average response delay of service requests obtained by the algorithms, the X-axis represents the number of VRCs that support each application service, and the Y-axis represents the minimum average response delay of service requests obtained by each algorithm.

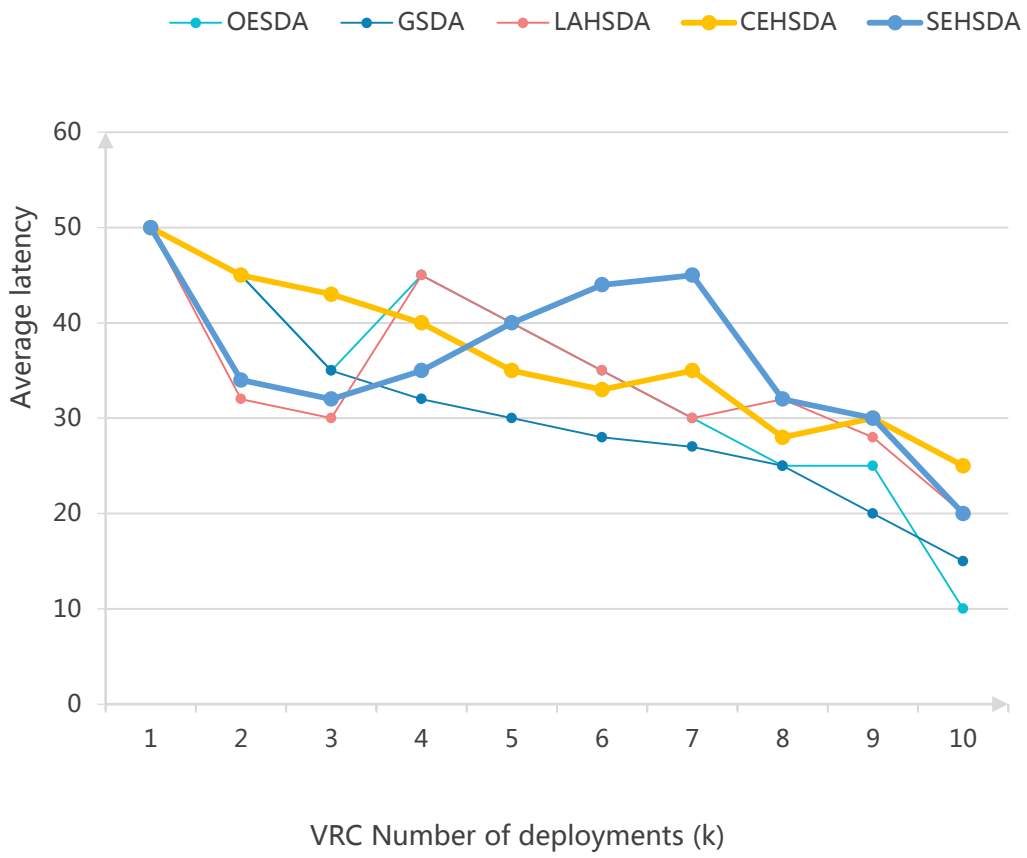


Figure 2. The average response time of the service request obtained

As shown in Figure 2, as the number of VRCs supporting each application service increases, the average response delay of service requests obtained by each algorithm decreases monotonically, because more VRCs support each application service, which can be reduced The average distance between the mobile device and the edge server that provides services for each application in the edge network, which leads to lower network transmission delay, and more edge servers to support the same application can provide more for the application Service resources, further reducing the average processing delay of service requests. It can be clearly seen from the figure that OESDA can get the best solution, while SEHSDA can get a near-optimal solution and is better than LAHSDA and CEHSDA. Experimental results also show that CEHSDA performs better than LAHSDA, and both are better than GSDA.

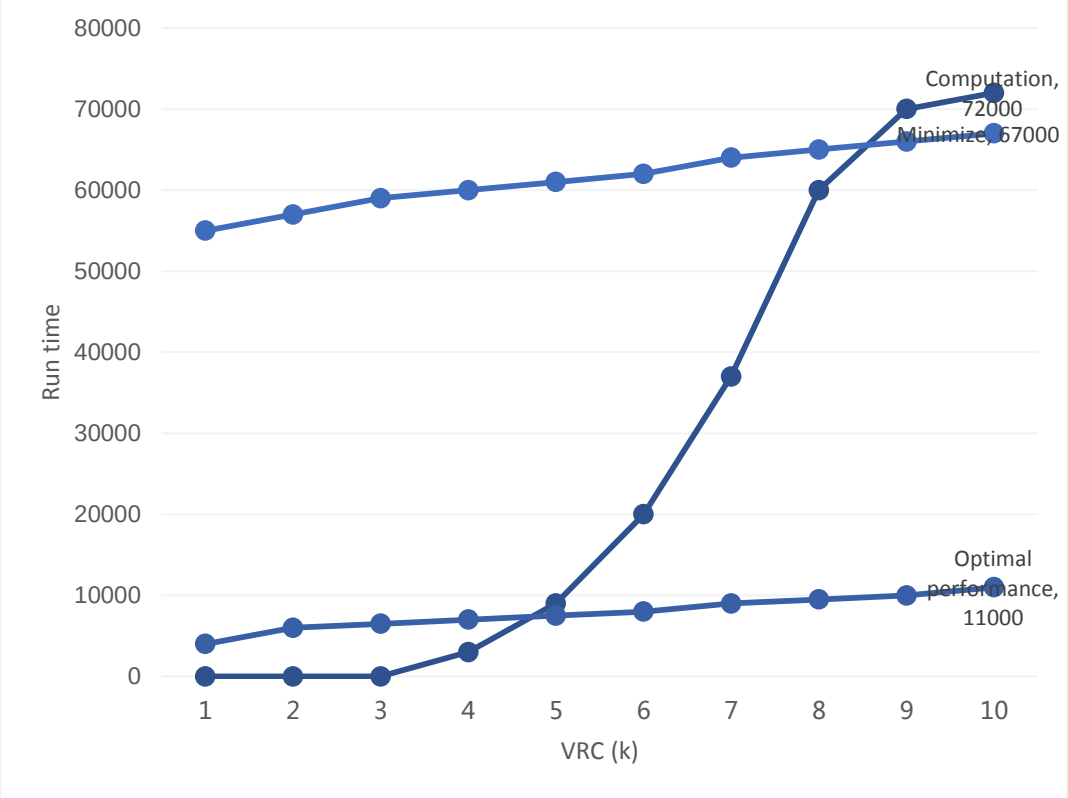


Figure 3. The comparison between the running times of the various algorithms

Figure 3 shows a comparison between the running time of each algorithm. As can be seen from the figure, as the number of VRCs supporting each application continues to increase, the computational complexity of OESDA increases exponentially, which is the complexity of all algorithms. The highest plan. Although SEHSDA's performance in minimizing the average response time is the algorithm closest to the optimal performance achieved by OESDA, its computational complexity is a little higher than that of CEHSDA. Compared with LAHSDA and GSDA, CEHSDA has reached the same conclusion.

4.2. Total Cost of Service Configuration

A good application configuration scheme not only ensures a low average response delay for service requests, but also needs to consider the cost of providing service configuration in the edge network. This chapter further analyzes the optimization performance of the application configuration schemes proposed by all algorithms in terms of service configuration costs.

In order to adopt different network topologies, the performance comparison results of GSDA, LAHSDA, CEHSDA and SEHSDA in minimizing the request response delay. It can be seen that the average response delay of requests obtained by the service configuration schemes proposed by all algorithms continues to rise with the growth of the network scale. Moreover, the experimental results also show that under each network topology setting, the SEHSDA algorithm can obtain the best performance in terms of reducing the average response time compared with GSDA, LAHSDA and CEHSDA. At the same time, it can also be seen that the average request response delay obtained by SEHSDA has increased slightly with the increase of the edge network size, but the

fluctuation of LAHSDA is much larger than other algorithms, especially when the network scale is large, LAHSDA is in these four the performance of these algorithms is the worst, as shown in Table 3.

Table 3. Network topology Settings table

Topological graph	number of nodes	Number of links
Spiralight	15	16
Sago	18	17
Noel	19	25
Shentel	28	35
Missouri	67	83

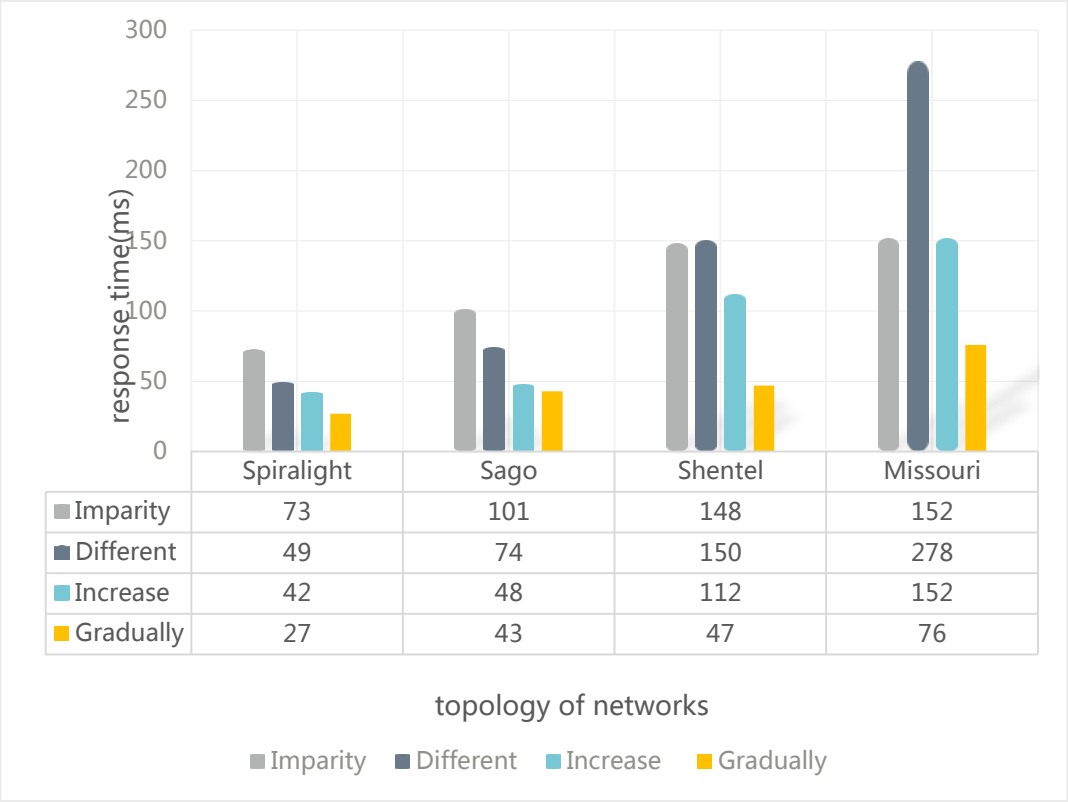


Figure 4. The comparison between the running times of the various algorithms

As shown in Figure 4, each application service will configure 3 VRCs in the edge network to support the service, and the types of different application services will gradually increase, and then compare the four algorithms. From the experimental results, it can be seen that with the increase of the types of different application services, the average request response delay obtained by the VRC configuration scheme proposed by Imparity has a very small increase, which is far better than other algorithms. It can also be seen from the figure that when the number of types of different application services does not exceed 3, the performance of Different is similar to that of Imparity, almost parallel. At the same time, the performance of Increase and Graduately are similar. However, when the number of types of different application services is not less than 4, the average request

response latency obtained by Different, Increase and Graduately is increasing rapidly. More importantly, the results obtained by Imparity and Graduately are similar to each other, and the effect of Different is much worse.

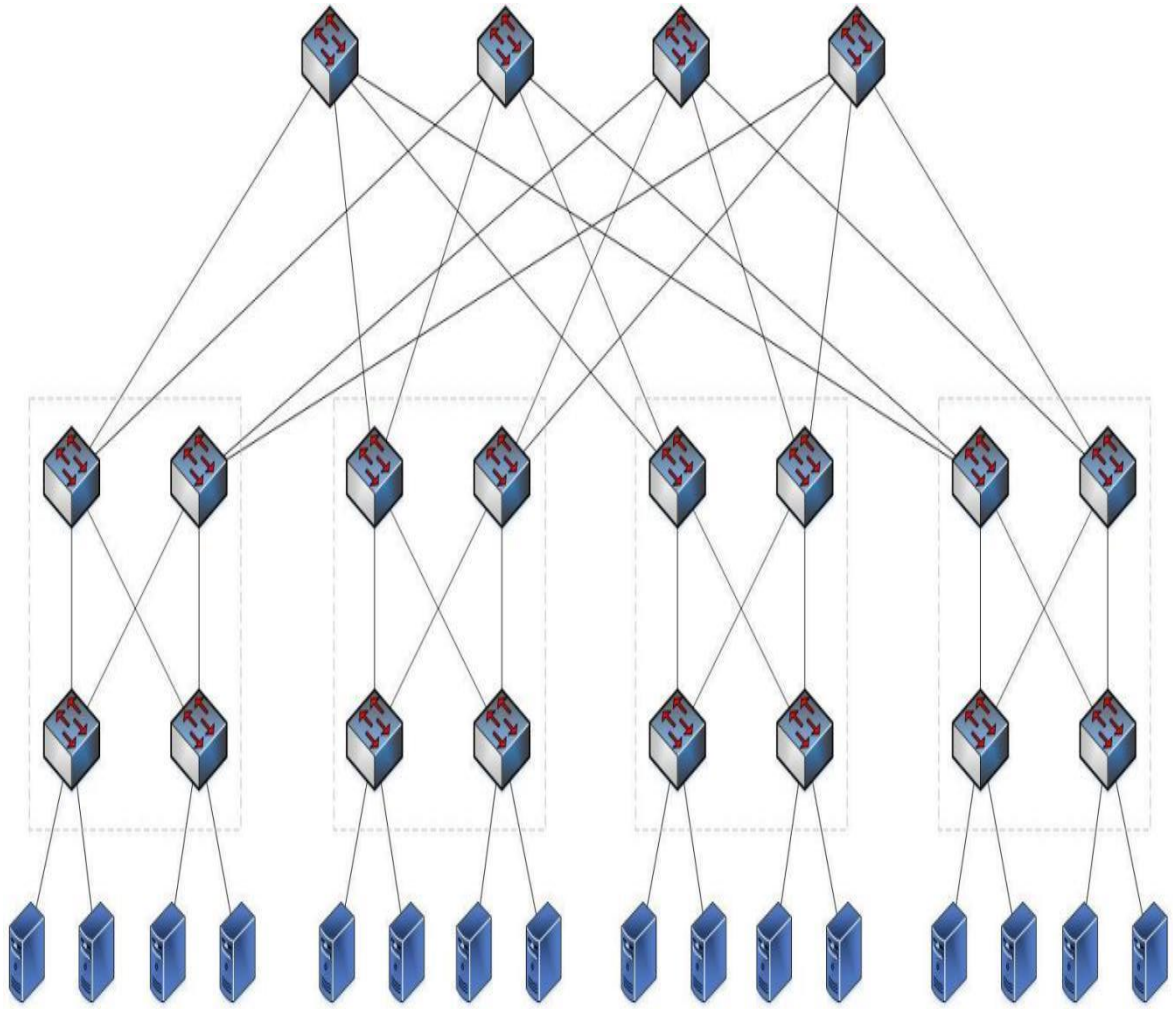
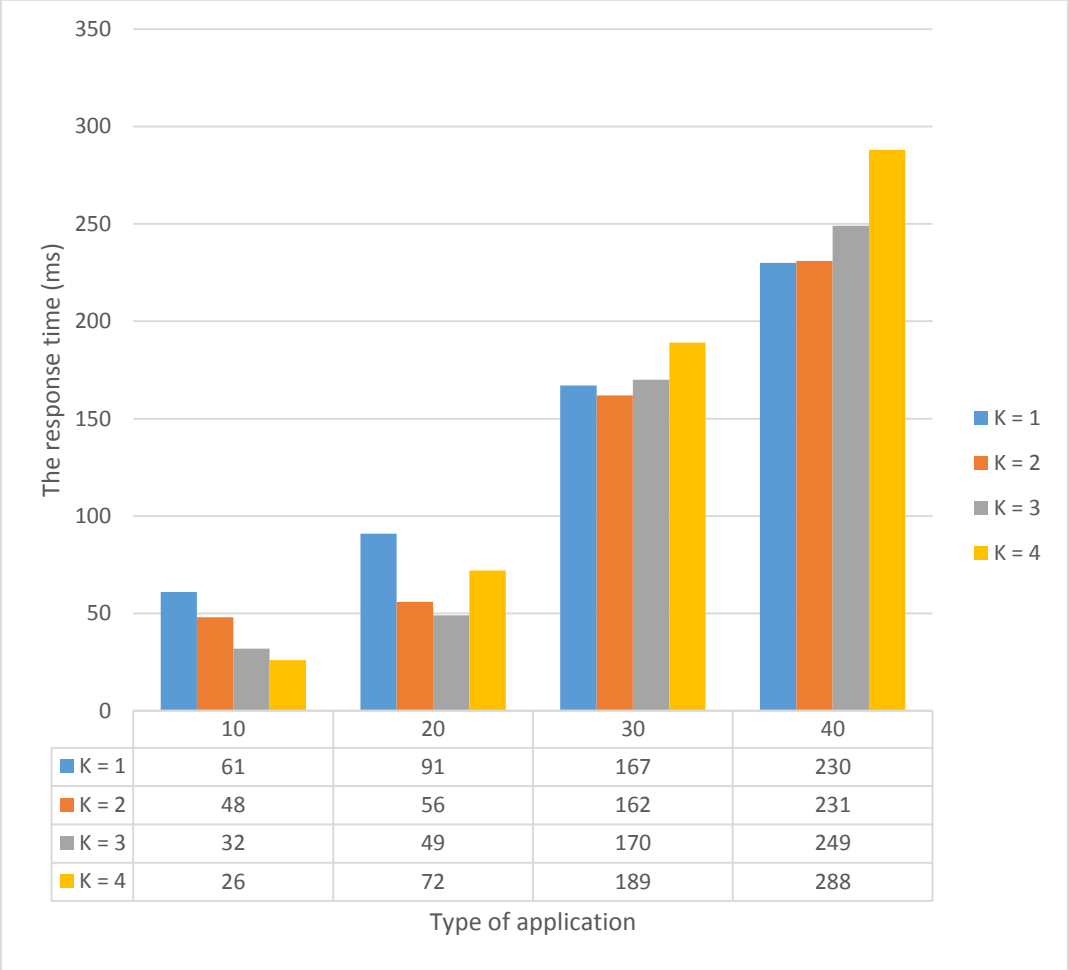


Figure 5. Fat-Tree network topology(From https://blog.csdn.net/baidu_38634017/article/details/88381721)

As shown in Figure 5, we can see from the figure that the Fat-Tree topology mainly has three layers of switches, from bottom to top they are edge switches, aggregation switches, and core switches. All switches in Fat-Tree have the same number of ports, and there was k. The switches at the edge layer and the aggregation layer are connected to each other and divided into different pods, as shown by the dotted line in the figure, and the number of pods is also k. In each pod, half of the ports at the edge layer of the switch are connected to the server and other aggregation layer switches in the pod, and the aggregation layer switches are similarly connected to the edge layer switches and the core layer switches. The core layer switch is connected to each pod in the network, and the number of edge layer switches and aggregation layer switches in the pod is equal and both are k/2. Therefore, the total number of switches in the network is k5/4, and the number of servers Is k3/4.



	10	20	30	40
K = 1	61	91	167	230
K = 2	48	56	162	231
K = 3	32	49	170	249
K = 4	26	72	189	288

Figure 6. Further analysis of different types of application services

As shown in Figure 6, when the number of different application service types becomes larger, the average request response delay obtained by the SEHSDA algorithm increases exponentially for the number of VRCs that each group supports each application service. This is because there are more configurations. Different types of application services to the edge network means that more service requests will need to provide service resources, which will lead to a shortage of available service resources in the edge server, an increase in the average service delay of requests, or more requests to pass longer the network delay is passed to the cloud computing center for service. Note that when the types of different application services are equal to 10, more VRCs supporting each application service will result in less average response time. However, when the number of different application service types becomes larger, more VRCs to support each application service may lead to a longer average request response delay.

After finding the best application server that provides the required service for all mobile device requests, OESDA uses a formula to calculate the average response delay for all service requests to obtain the service. Then OESDA continuously updates and records the currently obtained VRC configuration plan corresponding to the lowest average response delay of the request. After traversing all service configuration plans, OESDA will definitely be able to obtain the best VRC configuration plan for each service, such as Table 4 shows.

Table 4. Obtain the best VRC configuration for each service

	Construction stage	Cost management content	Calculation basis
Investment decision	Project proposal, feasibility studybook	Prepare investment estimates	Investment estimation indicatorsdata
design phase	initial design	Preparation of the overall design budget	Preliminary design drawings
Bidding stage	Bidding	Compile the block price	Construction drawings
construction stage	Contract/Project Implementation	Control cost, stage settlement	Control according to the contract price
Completion acceptance stage	Completion acceptance	Completion settlement / final accounts	Completion settlement (final accounts) and other documents

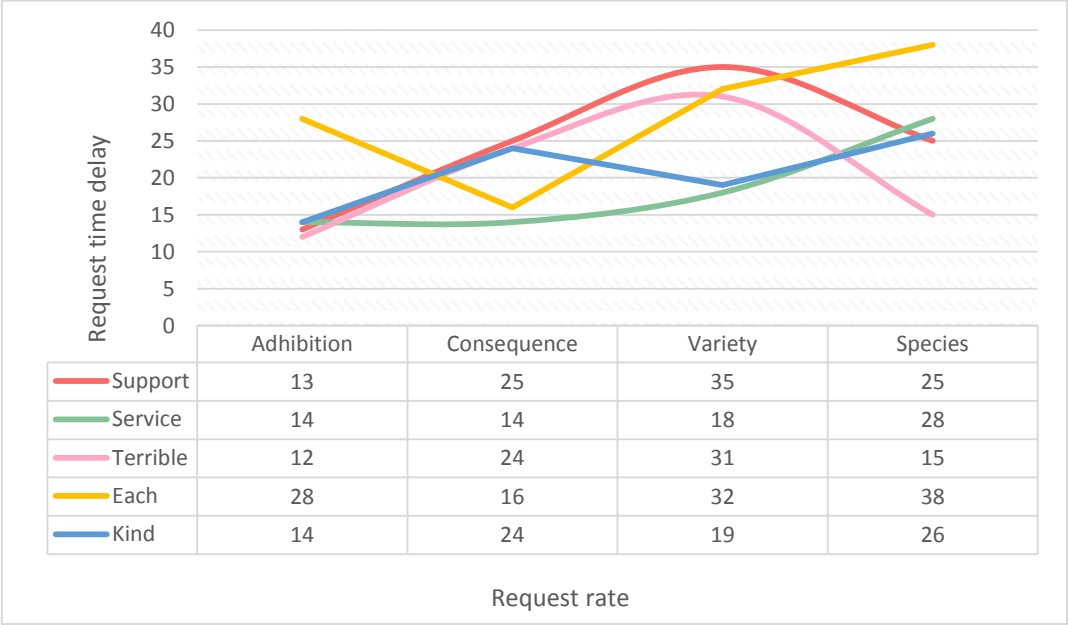


Figure 7. The request is passed to the cloud computing center

As shown in Figure 7, when the number of different application service types is equal to 20, configuring 4 VRCs to support each application service will lead to worse results than using 2 and 3 VRCs to support each application service. Moreover, when the number of different application service types was changed from 20 to 40, the result obtained by SEHSDA was completely reversed compared to before. The reason is that although more VRCs are deployed in the edge network, more edge servers can be obtained to support each application service, thereby greatly reducing the average response delay of application services, but as the number of different application types increases, The VRC hosted by each edge server will grow rapidly, which will quickly cause a large

amount of resource consumption in each edge server, and cause a long request processing delay or a large number of service requests to be passed to the cloud computing center, resulting in a long time network delay.

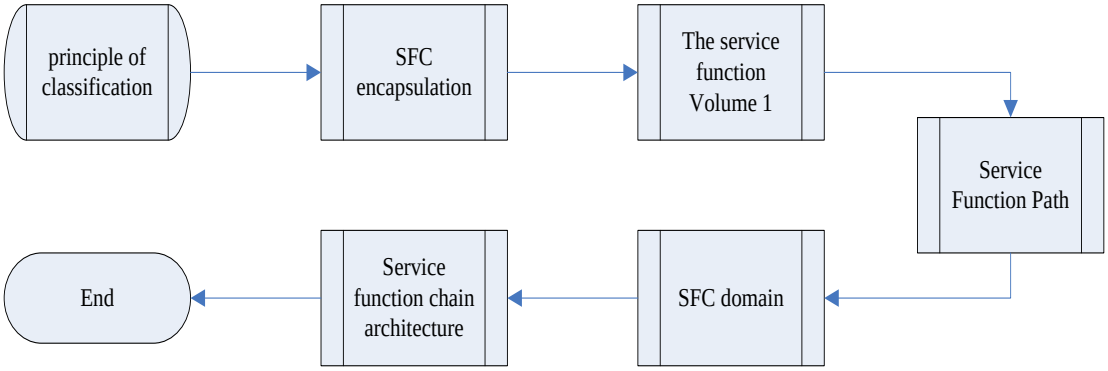


Figure 8. The architecture of the service function chain

As shown in Figure 8, the SFC classifier directs the network flow that meets the classification criteria to a service function path and forwards it to the requested service function. The classification function is executed by a classification function, and the initial classification occurs at the entrance of the service function chain domain. The granularity of classification depends on the capabilities of the classifier and the strategy of the service function chain. For example, coarse-grained classification directs all packets at the port that obey a certain service function chain strategy to a certain service function path; fine-grained classification adds rule matching on the basis of the service function strategy, and then directs the group to Another service function path. After the classifier is classified, it will install a suitable service function chain package for the data, and select or create a suitable service function path for processing.

5. Conclusion

Based on the mobile edge computing architecture, this paper proposes an edge network service model that provides application services for mobile users. It analyzes the optimal deployment of edge servers and proposes corresponding solutions, puts forward the optimal configuration of application services in edge networks and corresponding solutions, and studies multiple modes under limited edge network resources. However, mobile edge computing technology still faces many challenges when applied in actual scenarios, for example, how to intelligently route various types of request data in the edge network in real time, so as to maximize the throughput of the edge network, and how to be more efficient The application data of various different levels of the same application service is reasonably distributed among mobile devices, edge servers and cloud computing centers, so that the user's application experience is smoother and more comfortable, especially when considering multiple mobile devices that can share resources. In the case of equipment and multiple edge servers, how to perform efficient resource scheduling requires further in-depth research. I hope that I can solve the above problems one by one in the future experimental research, and can truly contribute to the resource allocation problem of the data center.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] *Zhang Q, Gui L, Hou F, et al. Dynamic Task Offloading and Resource Allocation for Mobile Edge Computing in Dense Cloud RAN. IEEE Internet of Things Journal, 2020, 7(4):3282-3299. https://doi.org/10.1109/JIOT.2020.2967502*

[2] *Yuan Q, Chen B, Luo G, et al. Integrated route planning and resource allocation for connected vehicles. China Communications, 2021, 18(3):226-239. https://doi.org/10.23919/JCC.2021.03.018*

[3] *Yang L, Xu G, Ge J, et al. Energy-Efficient Resource Allocation for Application Including Dependent Tasks in Mobile Edge Computing. KSII Transactions on Internet and Information Systems, 2020, 14(6):2422-2443. https://doi.org/10.3837/tiis.2020.06.006*

[4] *Jimoh O D, Lukman A, Oyetunde A O, et al. A Vehicle Tracking System Using Greedy Forwarding Algorithms for Public Transportation in Urban Arterial. IEEE Access, 2020, 8(2020):191706-191725.*

[5] *Fang W, Ding S, Li Y, et al. OKRA: optimal task and resource allocation for energy minimization in mobile edge computing systems. Wireless Networks, 2019, 25(5):2851-2867. https://doi.org/10.1007/s11276-019-02000-y*

[6] *XF Zhu, ZH Zhang, YL Wang. A dynamic resource allocation strategy in mobile edge computing environment. Computer Engineering and Science, 2019, 041(007):1184-1190.*

[7] *Li L, Liu H. Video Stream Session Migration Method Using Deep Reinforcement Learning in Cloud Computing Environment. Wireless Communications and Mobile Computing, 2021, 2021(1):1-10. https://doi.org/10.1155/2021/5579637*

[8] *Ni L, Zhang J, Jiang C, et al. Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets. IEEE Internet of Things Journal, 2017, PP(5):1-1. https://doi.org/10.1109/JIOT.2017.2709814*

[9] *Sun Y, Li Y, X Chen, et al. Optimal defense strategy model based on differential game in edge computing. Journal of Intelligent and Fuzzy Systems, 2020, 39(5):1-11. https://doi.org/10.3233/JIFS-179919*

[10] *Yang C, Lou W, Liu Y, et al. Resource Allocation for Edge Computing-Based Vehicle Platoon on Freeway: A Contract-Optimization Approach. IEEE Transactions on Vehicular Technology, 2020, PP(99):1-1.*

[11] *Peng H, Xu Y. Research on Resource Allocation Strategy of PaaS Platform. Journal of information technology research, 2019, 12(1):63-76.*

[12] *Saifeng Z. Research on caching and data real-time allocation virtual technology of cloud computing data center. Agro Food Industry Hi Tech, 2017, 28(1):1074-1078.*

[13] *Guo Y, Liu F, Xiao N, et al. Task-Based Resource Allocation Bid in Edge Computing Micro Datacenter. Computers, Materials and Continua, 2019, 58(2):777-792. https://doi.org/10.32604/cmc.2019.06366*

[14]Yang T, Shi X, Li Y, et al. Workload Allocation Based on User Mobility in Mobile Edge Computing. Journal on Big Data, 2020, 2(3):105-115. https://doi.org/10.32604/jbd.2020.010958

[15]J Choi. Opportunistic NOMA for Uplink Short-Message Delivery With a Delay Constraint. IEEE Transactions on Wireless Communications, 2020, 19(6):3727-3737.

[16]Deng L, Yang P, Liu W, et al. NAAM-MOEA/D-Based Multitarget Firepower Resource Allocation Optimization in Edge Computing. Wireless Communications and Mobile Computing, 2021, 2021(2):1-14. https://doi.org/10.1155/2021/5579857

[17]You Z, Lu I T. Cross-layer design benchmark for throughput maximization with fairness and delay constraints in DCF systems. Physical Communication, 2018, 28(JUN.):69-77. https://doi.org/10.1016/j.phycom.2018.03.005

[18] D Lin, Tang Y. Edge Computing-Based Mobile Health System: Network Architecture and Resource Allocation. IEEE Systems Journal, 2019, PP(99):1-12.

[19]Alabady S A. Saturation Throughput and Delay Performance Evaluation of the IEEE 802.11g/n for a Wireless Lossy Channel. Iraqi Journal for Electrical And Electronic Engineering, 2018, 14(1):51-64. https://doi.org/10.37917/ijeee.14.1.6

[20]Zhang Y, Di B, Zheng Z, et al. Distributed Multi-cloud Multi-access Edge Computing by Multi-agent Reinforcement Learning. IEEE Transactions on Wireless Communications, 2020, PP(99):1-1.

[21]Wei Y, Wang Z, Guo D, et al. Deep Q-Learning Based Computation Offloading Strategy for Mobile Edge Computing. Computers, Materials and Continua, 2019, 59(1):89-104.

[22]Zhang G, Zhang S, Zhang W, et al. Joint Service Caching, Computation Offloading and Resource Allocation in Mobile Edge Computing Systems. IEEE Transactions on Wireless Communications, 2021, PP(99):1-1.

[23]Zhang Y, Lo Y H, Shum K W, et al. New CRT sequence sets for a collision channel without feedback. Wireless Networks, 2017, 25(4):1697–1709. https://doi.org/10.1007/s11276-017-1623-x

[24]Liu J, Guo S, Liu K, et al. Resource Provision and Allocation Based on Microeconomic Theory in Mobile Edge Computing. IEEE Transactions on Services Computing, 2020, PP(99):1-1.

[25]Aung C Y, Ho W H, Chong P. Store-Carry-Cooperative Forward Routing with Information Epidemics Control for Data Delivery in Opportunistic Networks. IEEE Access, 2017, 5(99):6608-6625.

[26]Alsaffar A, Hung P, Huh E N. An Architecture of Thin Client-Edge Computing Collaboration for Data Distribution and Resource Allocation in Cloud. International Arab Journal of Information Technology, 2017, 14(6):842-850.

[27]Bhandari S, Zhao H P, H Kim, et al. An Optimal Cache Resource Allocation in Fog Radio Access Networks. Journal of Internet Technology, 2019, 20(7):2063-2069.

[28]Fan Y, Wang L, Wu W, et al. Cloud/Edge Computing Resource Allocation and Pricing for Mobile Blockchain: An Iterative Greedy and Search Approach. IEEE Transactions on Computational Social Systems, 2021, PP(99):1-13.

[29]Masadeh A, Salameh H B, Abu-El-Haija A. Design and Simulation of Spectrum Access and Power Management Protocol for Dynamic Access Networks. International Arab Journal of Information Technology, 2020, 17(4A):588-597. https://doi.org/10.34028/iajit/17/4A/2

[30]Chen M, Miao Y, Gharavi H, et al. Intelligent Traffic Adaptive Resource Allocation for Edge Computing-Based 5G Networks. IEEE Transactions on Cognitive Communications and Networking, 2019, PP(99):1-1.