Hindawi
Complexity
Volume 2020, Article ID 8936064, 11 pages
https://doi.org/10.1155/2020/8936064

Research Article

Resource Management Framework Based on the Stackelberg Game in Vehicular Edge Computing

Guang-Shun Li,[1,2] Ying Zhang,[1] Mao-Li Wang,[1] Jun-Hua Wu,[1] Qing-Yan Lin,[1] and Xiao-Fei Sheng[1]

[1]*School of Information Science and Engineering, Qufu Normal University, RiZhao 276800, China*
[2]*Department of Computer, The Hong Kong Polytechnic University, Hong Kong, Hong Kong*

Correspondence should be addressed to Jun-Hua Wu; shdwjh@163.com

Received 17 September 2019; Accepted 4 November 2019; Published 20 January 2020

Guest Editor: Xuyun Zhang

With the emergence and development of the Internet of Vehicles (IoV), quick response time and ultralow delay are required. Cloud computing services are unfavorable for reducing delay and response time. Mobile edge computing (MEC) is a promising solution to address this problem. In this paper, we combined MEC and IoV to propose a specific vehicle edge resource management framework, which consists of fog nodes (FNs), data service agents (DSAs), and cars. A dynamic service area partitioning algorithm is designed to balance the load of DSA and improve the quality of service. A resource allocation framework based on the Stackelberg game model is proposed to analyze the pricing problem of FNs and the data resource strategy of DSA with a distributed iteration algorithm. The simulation results show that the proposed framework can ensure the allocation efficiency of FN resources among the cars. The framework achieves the optimal strategy of the participants and subgame perfect Nash equilibrium.

1. Introduction

With the emergence of various Internet of Vehicles (IoV) applications, such as camera sensor data exchange, driving behavior analysis, voice recognition, real-time traffic information updates, and software downloads, a new architecture that can achieve ultralow delay and high throughput is highly required [1]. However, current wireless communication technologies cannot meet the ever-increasing service quality requirements for the following two reasons. As vehicles are densely deployed in some urban sections, complex and intensive vehicle services are formed, which increases the difficulty of data services. Additionally, current communication technologies cannot support a large number of user terminals. In cellular networks, the spectrum efficiency drops drastically with increasing user density [2, 3]. To meet the demand for data computing services, a large number of large-scale data centers have been deployed. In addition, cloud computing has recently been proposed to provide flexible and efficient services to data service subscribers [4]. In cloud computing, the data service agent can organize a shared pool of configurable computing resources (such as servers, storage, networks, applications, and services), which can be easily accessed by data service subscribers on demand [5].

However, the distance between mobile vehicles and remote cloud servers may cause a large network transmission delay and create considerable overhead, which is intolerable for applications requiring real-time interaction and high mobility requirements [6, 7]. Accordingly, it is beneficial and necessary to bring the cloud closer to the users. In this case, a new architecture and technology called MEC emerged, pushing cloud services to the edge of wireless networks and providing services close to mobile vehicle terminals. In MEC, the network edge can run in an environment isolated from the rest of the network and create access to resources in the local neighborhood. Moreover, in IoT, fog computing was proposed by Cisco as a promising solution. In fog computing, multiple low-power computing devices, commonly referred to as the FNs, at the edge of

networks are deployed to offload the data computing services from the cloud [8]. With the properties of small scale, low construction cost, and mobility support, the FNs are generally deployed much closer to the data service subscribers; thus, the network latency of accessing cloud computing services can be greatly reduced, enabling MEC to provide fast interactive responses and location-aware services in data services [9]. With different purposes and preferences, data service subscribers at the network edge can receive data services from the FNs in the neighborhood.

In this paper, MEC technology is introduced into the IoV to form vehicular edge computing (VEC). In the VEC network, the concept of network virtualization is also applied. As the large number of FNs and their computing resources are invisible to the cars, the car can only contact and purchase data services from the DSA. Therefore, there is a virtualized network between the DSA and the car. When service requests are received from all cars, each DSA can collect computing resources from the FN and provide virtual data services to the car. Thus, the computing resource can be efficiently and effectively utilized by nearby cars. Each car is a data service subscriber and needs to apply for data services from the DSA.

An edge computing network can consist of a large number of FNs deployed by different DSAs at different locations to provide various data services and applications to the cars. When cars can choose their DSA as well as the corresponding FN to further enhance their quality of experience, how to form a data service framework that is more efficient and meets the highest revenues of all participants is still an open problem. In this paper, we focus on the service radius of each DSA and the benefits of each layer, proposing an efficient data resource management framework. In this framework, combined with cars' highly mobile characteristics and motion characteristics, we divided the service area of DSAs according to the density and speed of the car. Then, we developed a Stackelberg game to simulate the interaction between the FN and the DSA. The FN determines their service price first, and the DSA then decides to purchase the optimal number of computing resource blocks (CRBs). Once the price of the FNs and the purchased resources of the DSAs have been obtained, the DSA offloads this data demand to the corresponding FN. However, if there are not enough CRBs available in the FN to meet the data service requirements of all DSAs, some DSAs will be served by remote data centers that are remote from the DSA.

The rest of the paper is organized as follows. Related work is described in Section 2. We introduce the system framework and define the problem in Section 3. In Section 4, we propose a dynamic service area partitioning algorithm to divide the service area of the DSA. In Section 5, we use the Stackelberg game to analyze the two layers of interaction in the framework. Section 6 shows the experimental evaluation based on a real dataset. Finally, a conclusion is drawn in Section 7.

2. Related Work

In a lot of studies, fog computing has been advocated to be the promising future of the cloud. Ahlgren et al. [10] studied the concept of mist computing, aiming to distribute the cloud and its benefits deeply into the network. Yannuzzi et al. [11] considered the requirements of mobility, scalability, reliable control, and actuation in some challenging scenarios of IoT to show the benefits and significance of fog computing. A recent survey on the emerging 5G network edge cloud architecture can be found in [12]. Taleb et al. have analyzed the MEC reference architecture and main deployment scenarios and conducted an overview of the current standardization activities. Li et al. [13] proposed an intermediary framework, where there exists an intermediary between multiple cloud providers and users. The intermediary first rents the cloud service from cloud providers and then provides streaming processing service to users with low cost and delay. In [14], a highly localized IoT-based cloud computing model was proposed. Aura allows mobile clients to create ad hoc and flexible clouds using the IoT and other computing devices in the nearby physical environment. Li et al. [15] have presented a user-oriented improved spectral clustering scheduling algorithm to solve the problem of resource scheduling and improve the satisfaction of users. And in [16], the methods of fuzzy clustering were combined with particle swarm optimization to divide the resources, which improves user satisfaction and the efficiency of resource scheduling. However, mobile cloud computing also requires a high quality of network connections with remote infrastructures. Therefore, on the basis of these studies, our work is to build a layer of edge network between the cloud and the end user, to provide data services for the car, and use the car as the end user, so the car is also a part of the edge. Our edge node layer can be closer to the end user; instead of sending the information to the remote server, some data services can be provided directly by edge nodes to improve service efficiency. In [17], a Stackelberg game theoretic model was shown for dynamic bandwidth allocation between virtual networks. Wu et al. [18] considered a Stackelberg game between data center and buses in the smart city, where each bus collects data along its route and competes with other buses for the reward forwarding to the data center. In the game, following the proposed heuristic algorithm, the Stackelberg equilibrium is shown to be achieved where the data center and each bus are able to reach maximum utility. Wang et al. [19] modeled the interaction between the monopolistic data center operator and the customers as a Stackelberg game. In the game, the pricing strategies of the monopolistic data center operator and the corresponding behavior of data service customers are detailedly analyzed in both homogeneous and heterogeneous customer scenarios.

In recent years, in many studies, the combination of resource scheduling and cloud computing in the Internet of Vehicles has achieved research results. Shiraz et al. [20] have proposed thematic taxonomy of current DAPFs, reviewed current offloading frameworks by using thematic taxonomy, and analyzed the implications and critical aspects of current offloading frameworks. In [21], a local roadside cloud-based network is proposed to deal with traffic-related data, which is coincident with the goal of fog computing. A vehicular fog computing (VFC) architecture

is proposed in [22], in which vehicles with redundant resources are used as the computational infrastructures and the burdens of congested resource-limited vehicles are relieved. Alamer et al. [23] modeled a CVCC network by a two-phase heterogeneous public good game and then investigated the influence of different incentive mechanisms and the structure of a complex network describing the vehicles' connectivity on the vehicles' investment rate. Kumar et al. [24] have discussed the use of vehicular delay tolerant network technologies for MEC targeting mainly at smart grid applications. Salahuddin et al. [25] proposed a novel roadside unit (RSU) cloud, a vehicular cloud, as the operational backbone of the vehicle grid in the Internet of Vehicles (IoV). The architecture of the proposed RSU cloud consists of traditional and specialized RSUs. Kim et al. [26] considered an innovative RSU deployment framework, which is a well-balanced combination of three different approaches: deploying RSUs on static locations, public mobile transportation, and fully controllable vehicles owned by the local government. Zhang et al. proposed a cooperative fog computing-based intelligent vehicular network for dealing with big IoV data, and they further discussed mobility control and distributed computation and storage [27]. However, our research is quite different from theirs. Although we both mentioned the features such as using edge computing to solve the problem, our management framework highlights the new features such as taking into account the impact of the vehicle's mobile characteristics on the service provided by the edge nodes and making a prejudgment of the service capabilities to provide low-latency communication and more context awareness.

3. System Framework

The car needs data service during the driving process, and each car can deliver data to the FN at the edge of the network. Each DSA selects an FN to provide cars with the required data services, as shown in Figure 1. Such a three-layer edge network is the main core framework of this paper. DSA is located in the middle layer, which serves the lower car and manages the upper FN through a connecting car and the FN. We define the unit number of computing resources that can be distributed by each FN as the CRB, each of which can provide computing service at the rate of μ. The physical data transmission network between FNs and cars satisfies the SecondNet topology, where the network facilities can provide guaranteed quality of service (QoS) for the DSSs. Accordingly, to reduce the risk of potential network congestion and achieve real-time fast-response interaction, each DSA tries to offload the data service submitted by a car to the FN. However, as the car cannot have the authorization to access the CRBs directly, the cars are required to receive the virtualized services from the DSAs, and with the management of DSAs, the CRBs of the FNs can finally be allocated to the cars.

The system architecture is shown in Figure 1. FN stands for the fog server, DSA is the multidata agent, and car stands for the data service subscribers (see Figure 1).

4. Service Area Partitioning Algorithm

To enable the car to apply for services at any time and reduce the service response time, it is necessary to divide the service area of each DSA. The service area of a DSA is a circle with a radius of R_{ref}. By default, all cars in this circle are served by the corresponding DSA. Each car has a corresponding DSA, and the service area of all DSAs is added to M. Each car can communicate with an FN through a DSA or with vehicles in different service areas through mmWare or other networks. One FN is responsible for one DSA, and each DSA can communicate directly with the corresponding FN. Because the traffic conditions on different roads are different and the driving of vehicles is random, the density of vehicles in the service area of each DSA is different, which leads to the unbalanced load of the DSA. If there are too many cars requesting data services within a DSA service area, the response time of service will be too long and even the quality of service will be affected. Therefore, we propose a dynamic service area partitioning algorithm that helps us adjust the service area radius R_{ref} according to the car driving behavior in the circle. We mainly consider three factors: speed factor, location factor, and server idle resources with the algorithm. Each car is connected with mmWare, and each node shares its own information with hello messages.

4.1. Acquisition of Vehicle Motion Data

4.1.1. Interpolation Method. We obtained the speed of the car with the interpolation method in [28]. The interpolation method is a function value that uses a function $f(x)$ to know several points in a certain interval. If there is an appropriate specific function, other points in the interval can obtain an approximation of the function $f(x)$ with the value of the specific function. This is the interpolation method.

Let function $y = f(x)$ be defined on the interval $[a, b]$, and there exist a set of values $y_0, y_1, \ldots, y_n$ at a set of points $a \le x_0 < x_1 < \cdots x_n < b$; if there is a simple function $p(x)$,

$$p(x_i) = y_i, \tag{1}$$

then $p(x)$ is the interpolation function of $f(x)$, the point $x_0, x_1, \ldots, x_n$ is the interpolation node, and the interval containing the interpolation node $[a, b]$ is the interpolation interval.

The linear interpolation formula is

$$p(x) = a_0 + a_1 x. \tag{2}$$

Set the pixel coordinates of the rear wheel center point to the previous frame t_0 and the next frame t_1 to be (x_0, y_0) and (x_1, y_1), respectively. The pixel coordinates of the front wheel center point in the middle of the two frame images are (x, y), and the time t of the front wheel at this frame is

$$t = \frac{t_1 - t_0}{x_1 - x_0} \times (x - x_0). \tag{3}$$

The principle of calculating the speed of the video is

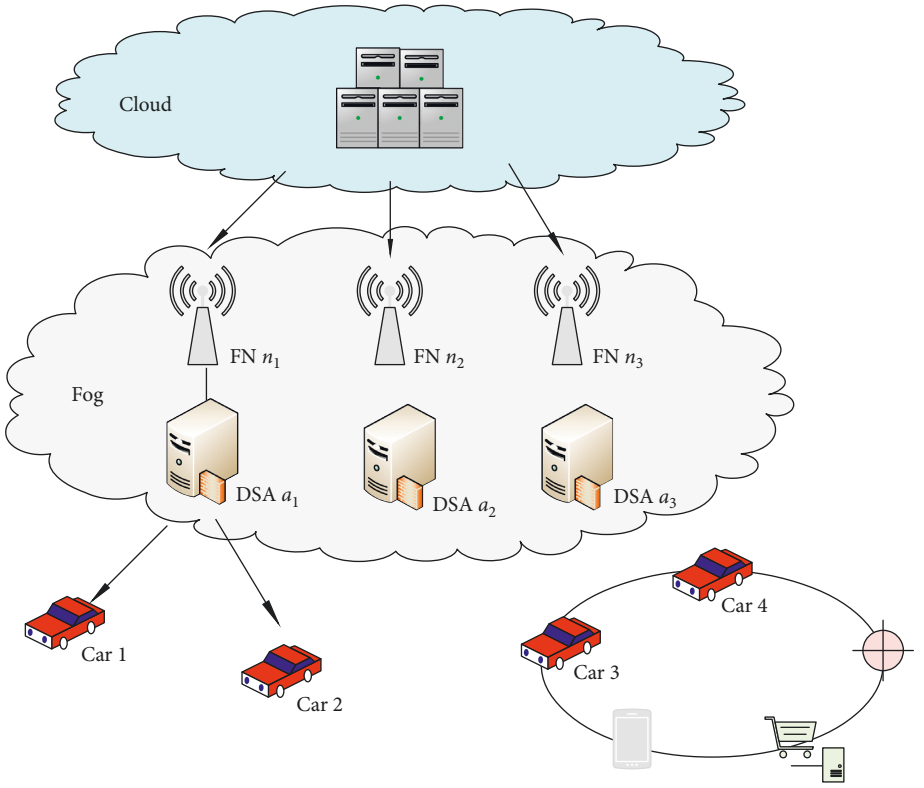

Figure 1: System framework graph.

$$v = \frac{\Delta d}{\Delta t}. \tag{4}$$

In general, the above formula is known as the Δd or Δt value of a variable, and then, another calculation method is used to obtain another variable value.

4.1.2. Time Interpolation Method to Obtain Speed. In this study, we calculate the speed of a car based on video images using the interpolation method. We adopted the most basic linear interpolation. The motion data of the car are obtained by using a video. In the video image, the captured target vehicle can find more than two feature points with known distances in the moving direction and take the actual distance between the two feature points as the scale length. The pixel coordinate values of each feature point in the image are combined with the interpolation principle to calculate the exact time when the vehicle passes a scale distance, finally obtaining an accurate vehicle speed.

Assume that the wheelbase L of the vehicle is set to the scale and the direction of driving is as shown. At 0 frames, the front wheel position is $X(0)_q$ and the rear wheel position is $X(0)_h$; in the n frame, the front wheel position is $X(n)_q$ and the rear wheel position is $X(n)_h$; in the $n+1$ frame, the front wheel is at the $X(n+1)_q$ position and the rear wheel is at the $X(n+1)_h$ position (see Figure 2).

If $X(n)_h = X(0)_q$, the target vehicle passes the distance L for exactly n frames, and the average speed is

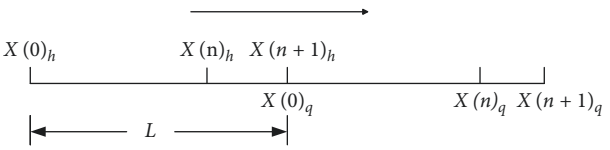

Figure 2: Schematic diagram of the time interpolation method.

$$v = \frac{L}{nt} \times 3.6. \tag{5}$$

When $X(n)_h < X(0)_q$, find the time difference value between $X(n)_h$ and $X(0)_q$ with the interpolation method:

$$\Delta t_1 = \frac{X(0)_q - X(n)_h}{X(n+1)_h - X(n)_h}. \tag{6}$$

The average velocity is

$$V(x) = \frac{L}{(n + \Delta t_1) \times t} \times 3.6. \tag{7}$$

4.2. Affecting Factors of Service Radius

4.2.1. Speed Factor (VF). Calculate $\mathrm{VF}(s,m)$ according to the time interpolation method:

$$\mathrm{VF}(s,m) = \frac{|v(m)| - \min_{y \in N_s} |v(y)|}{|v(y)|}, \tag{8}$$

where N_S represents a group of nodes in the neighborhood and $v(\,)$ represents the speed. A smaller VF indicates a lower velocity. Based on the weighted exponential moving average, the VF is updated periodically at an interval of 10 seconds:

$$\mathrm{VF}_i(s,m) \longleftarrow (1-\omega)\times \mathrm{VF}_{i-1}(s,m) + \omega \times \mathrm{VF}_i(s,m), \quad (9)$$

where $\mathrm{VF}_{i-1}(s,m)$ and $\mathrm{VF}_i(s,m)$ represent the previous and current values of VF, respectively. ω represents the influence value of the current speed on speed change, VF is initialized to 1, and ω is set to 0.7.

4.2.2. Location Factor (LF). The location factor is calculated as follows:

$$\mathrm{LF}(s,i) = \left| tv_{si} - 2\left(R_{\mathrm{ref}1} + R_{\mathrm{ref}2} + \cdots + R_{\mathrm{ref}i}\right)\right|. \quad (10)$$

LF indicates the position of the car from the edge of the current service area and tv_{si} indicates the distance traveled by the car s in area i. A smaller LF indicates that the car is about to leave the service area.

4.2.3. Server Idle Resources. The server idle resources is given as follows:

$$\begin{aligned} &\left(1-\delta_i\right) \le \delta_\Delta \\ \text{s.t}\quad &\sum_{i=1}^{R} 2\left(R_{\mathrm{ref}i}\right)\pi \ge M,\ \exists i,\ \forall R\left(1-\delta_i\right) \ge \delta_{\min}, \end{aligned} \quad (11)$$

where δ_i represents the resource occupancy of the server. All service areas add up to exceed M, and idle resources should be greater than the minimum value. Only by guaranteeing these two conditions can the server provide data resource services.

4.3. Service Area Partitioning Algorithm. If the car is fast or closer to the edge of the DSA's service area, then determine if it will take a short time to reach the next DSA. If so, if it is small enough, when the server is busy, the job cannot be queued first, and the car may soon reach the service area of the next DSA and be served by the next DSA. When a certain number is reached, it means that the car density decreased before the current DSA processed the job. There is no need to change R_{ref} in this situation. Only when the idle resource of the current DSA is less than the threshold, and the car within the service area can only provide the service by the DSA within a certain period of time will the radius of the service area change, and R_{ref} is changed 2 times. Conversely, when a DSA is idle, the job of the busy DSA is transferred to the DSA with idle server resources. If there are still more free resources, R_{ref} is changed to half of the original R_{ref}. The sum of the service areas of each DSA after the change must be equal to the original total area M (e.g., in Figure 3). The service area partitioning algorithm is as follows (see Algorithm 1).

5. Stackelberg Game Decision

5.1. Definition of the Stackelberg Game. The Stackelberg game is a game problem in which the decision-makers are in a master-slave relationship; that is, the status of game players belonging to the two decision-making layers is inequitable, and the players in the upper-layer game are more influential than the lower-layer players. It is a leader and follower relationship. The leader always takes the lead in making decisions, while the followers make the best decisions based on the strategy of the leader and other followers in the same layer.

A game model usually consists of three elements: the game player, strategy, and revenue. Because the Stackelberg game has two different decision-making layers, its players are divided into two categories and have their own strategic space and revenue.

Game player defines m leaders and n followers, respectively, and is represented by two sets $A = \{1, 2, \ldots, m\}$ and $B = \{1, 2, \ldots, n\}$.

Strategy defines the strategy combination of leader as $x = \{x_1, x_2, \ldots, x_n\}$ and the set as X and the strategy combination of follower $y = \{y_1, y_2, \ldots, y_n\}$ and the set as Y.

Revenue of leader i is expressed as U_{S_i}, and the revenue of follower j is expressed as U_{O_j}, where $i \in M$ and $j \in N$.

The above game model is defined as a Stackelberg game with multiple leaders and multiple followers. Assuming that the leader has made a decision, each follower who participates in the game plays a noncooperative game under this decision to maximize their own revenues; then, the best strategy for the follower under the leader strategy can be expressed as

$$\begin{aligned} S(x) = \Big\{ y^* = (y_1^*, y_2^*, \ldots, y_n^*) &: U_{O_j}\left(x, y_j^*, \ldots, y_{-j}^*\right) \\ &\ge U_{O_j}\left(x, y_j, \ldots, y_{-j}^*\right)\Big\}, \end{aligned} \quad (12)$$

where y_j^* represents the best strategy for follower j and y_{-j}^* represents the best set of strategies for followers other than j. When all the follower strategies satisfy the above formula, the optimal strategy space $y = (y_1^*, y_2^*, \ldots, y_j^*)$ is called a Nash equilibrium of the noncooperative game [29].

5.2. Stackelberg Game Analysis for Two-Layer Interaction. Assume that in a particular edge computing network, there are multiple FNs set to set $M = \{1, 2, \ldots, m\}$. The set of DSAs covered in this range is $N = \{1, 2, \ldots, n\}$, and FN competes for all DSAs. The price strategy of the FN_j node is p_j, $j \in M$, and the price strategy of all FNs is $p = (p_1, p_2, \ldots, p_m)$. The CRB requirement strategy of DSA_i is x_{ij}, $i \in N$, which means the quantity of CRB purchased by DSA_i at FN_j. We define $x_i = (x_{ij}, x_{-ij})$ as the CRB requirement strategy vector of DSA_i, where x_{-ij} represents the strategy of DSA_i at other FNs except for FN_j and $q = (q_1, q_2, \ldots, q_n)$ denotes the set of data requirement strategies for all DSAs.

The Stackelberg game mainly means that one player of the game first predicts the purchase amount of the other player to determine the price and the other player determines the purchase amount according to the price so that the determined price is considered by both parties to take care of the revenues of both parties. In our model, we consider DSA as the applicant for computing resources and as a follower in

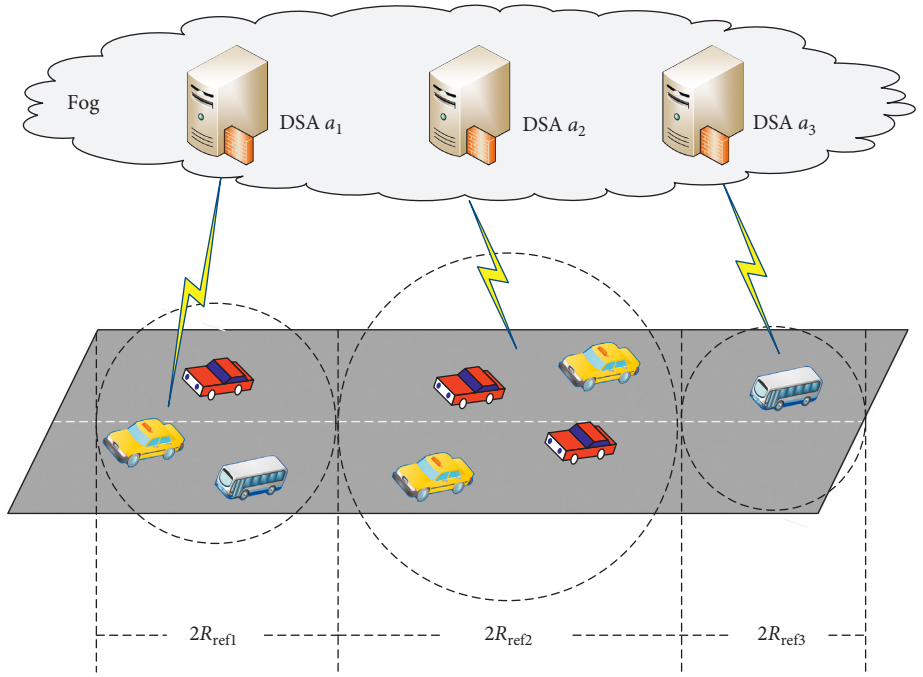

Figure 3: Schematic diagram of service region division.

Input: *The number of* DSA
The service radius for each DSA
The Car density k_n
According to the above input, bring the program to determine the radius, get the R_{ref}
if $((1-\delta_i) \le \delta_\Delta)$
  while $(k_n \ge n)$
  for $(s = 1, s \le k, s{+}{+})$
  if $(VF(s,m) \ge v_\Delta \parallel LF(s,m) \le l_\Delta)$
    $t_k = (LF(s,m)/VF(s,m))$
    if $(t_k \le t_\Delta)$
      counter +1
      if $(counter \ge 0.5k_n)$
        keep R_{ref}
      else
        join the DSA queue list
        if $(l_{th} \ge 0.75k_n)$
          $R_{ref} = 2R_{ref}$
else $((1-\delta_i) > \delta_\Delta)$
  move tasks from the adjacent DSA queue list to local completion
  if $(\delta_i' \ge \delta_\Delta)$
    $R_{ref} = (1/2)R_{ref}$
output R_{ref}

Algorithm 1: DSA service area partitioning algorithm.

the game model. The FN needs to provide computing services to DSA, which is the leader in the game model. The game between the FN and the DSA consists of two phases. In the first phase, different FNs first declare their price strategy p and broadcast the strategy to all DSAs. In the second phase, the DSA makes its own data resource strategy q based on the received price strategy vector p. After determining the price strategy and data resource strategy, the competition of the DSA and the FN constitutes a noncooperative game problem. The strategic combination of the DSA and the FN (p, q) is a solution to the Stackelberg game. Next, we formulate the utility of the DSA and the FN.

5.2.1. Utility of DSA. Each DSA independently selects the FN to develop a CRB demand strategy. The utility function of the

DSA is composed of two parts: the benefits and costs of providing data services for a car. The benefits are related not only to the number of CRBs but also to the satisfaction of different cars with CRBs of the same unit. The cost of the DSA includes the cost of purchasing the CRBs from the FN and the delay in the data service. The increase in price and the increase in data transmission delay caused by network congestion will lead to the DSA adjusting the data resource strategy. The utility of the DSA can be expressed by the following formula:

$$U_{Ni} = U_i\left(\sum_{j=1}^{m} x_{ij}\right) - \sum_{j=1}^{m} P_j\left(p_j x_{ij}\right) - \sum_{j=1}^{m} D_j\left(x_{ij}\right), \quad (13)$$

where $U_i(\sum_{j=1}^{m} x_{ij})$ is the total revenue earned by the DSA when car is served by the DSA, p_j is the price set by the FN, $P_j(p_j x_{ij})$ is the price paid by the DSA to the FN, and $D_j(x_{ij})$ represents the delay cost of the DSA's service to cars.

We only study flexible business flow. When the data service requested by the car is low, the marginal effect of the DSA based on data is very large, but the marginal effect decreases with the increase in data. Its benefit function can be described as an increasing concave function based on the total resource, which generally conforms to the trend of the logarithmic function. Therefore, we use the logarithmic function $U(x) = \alpha \log(1 + x)$ to represent the utility function of the DSA flexible business flow, where α is a constant greater than zero, related to the sensitivity to the data latency of the car, and x represents the total number of CRBs obtained by the DSA. Finally, the goal of each DSA is to choose its own optimal data resource strategy x_{ij}^*, which is $\arg\max U_{Ni}(x_{ij}, x_{-ij}, p_j, p_{-j})$, where x_{-ij} and p_{-j} indicate that other DSAs and FNs also choose the optimal data resource strategy and price strategy.

We assume that the workload of each DSA follows the Poisson arrival process. If the total load $Q_j \geq C_j$ of all DSAs in an FN, the network will be congested. Only when the load of all DSAs in the FN satisfies $Q_j < C_j$ can the effective transmission of data be guaranteed. Specifically, the load of FN is Q_j, and the delay cost function of DSA_i in FN_j can be expressed as

$$D_j(q) = \begin{cases} \dfrac{\beta_j}{C_j - Q_j}, & \text{if } Q_j < C_j, \\ \infty, & \text{if } Q_j < C_j, \end{cases} \quad (14)$$

where β_j is a constant related to data transfer technology.

5.2.2. Utility of FN. For FN, the total utility is the payment received from the DSA minus the transmission cost. We set c_{ij} to the transmission cost per unit CRB, and DSA_i is the service price per unit for the FN. Therefore, the utility of the FN can be expressed by the following formula:

$$U_{Mj} = \sum_{i=1}^{n}\left(p_j - c_{ij}\right)x_{ij}, \quad (15)$$

where $\sum_{i=1}^{n} p_j x_{ij}$ is the total revenue received by the FN from the DSA and $\sum_{i=1}^{n} c_{ij} x_{ij}$ is the total transmission cost estimated by the FN.

To obtain data services from the FN, the DSA needs to purchase a small amount of CRB from the FN to achieve satisfactory service, which comes at a price. Different DSAs have different service delay tolerances. When the upper limit of the service delay is high, the DSA only needs to buy a small amount of CRB to achieve satisfactory service. However, when the upper limit of the service delay is low, the DSA must purchase a large number of CRBs to ensure that the quality of service is improved and the service delay is within the tolerant area. In addition, the price of the service set by the FN will also affect the utility of the DSA. When the price is high, even if a large number of CRBs can improve the quality of data services, the DSA also needs to pay a large number of fees to the FN, so the benefits may not be satisfactory. How many resources does a car need to apply to complete its own tasks but also to ensure the quality of service without wasting resources? The FN provides data computing services to the DSA and needs to set a price that benefits the DSA. However, if the price is set too high, the DSA will reduce the number of CRB purchases or choose another FN, so it is necessary to predict the response of the DSA to determine the service price to maximize utility. Therefore, how the FN prices its resources to protect its revenue without losing user satisfaction is the key.

5.3. Proving the Existence of Nash Equilibrium Based on a Utility Function. According to the previous derivation, the utility function of DSA_i can be expressed as

$$U_{Ni}\left(x_{ij}, x_{-ij}, p_j, p_{-j}\right) = \alpha_i \log\left(1 + \sum_{j=1}^{m} x_{ij}\right) - \sum_{j=1}^{m} p_j x_{ij} - \sum_{j=1}^{m} \frac{\beta_j}{C_j - Q_j}. \quad (16)$$

According to the utility function of the DSA, the second derivative of U_{Ni} relative to x_{ij} is

$$\frac{\partial U_{Ni}}{\partial x_{ij}} = \frac{\alpha_i}{1 + \sum_{j=1}^{m} x_{ij}} - p_j - \frac{\beta_j}{C_j - Q_j}. \quad (17)$$

The result of finding the second-order partial derivative of the utility function U_{Ni} of the DSA is

$$\frac{\partial^2 U_{Ni}}{\partial^2 x_{ij}} = \frac{\alpha_i}{\left(1 + \sum_{j=1}^{m} x_{ij}\right)^2} - \frac{2\beta_j}{\left(C_j - Q_j\right)^3} < 0. \quad (18)$$

The utility function of the DSA is proved by a strictly concave function, so the Nash equilibrium point exists.

5.4. Solution of the Stackelberg Game Problem. Aiming at the characteristics of local sharing of decision information among players in the Stackelberg game model mentioned above, we solve the perfect Nash equilibrium of the subgame with the distributed iteration algorithm proposed in [30]; that is, each player can determine the best strategy only with local information. Assume that at time t, the price strategy of

the FN broadcast is $p(t)$. Based on the demand for data resources, DSA considers the price and service capabilities of the FN and adjusts its data resource strategy to maximize its utility. The rate of change in the data obtained by the DSA at each FN is proportional to the gradient of the utility function. The time from τ to $\tau + 1$ is defined as an iteration period $\Delta\tau$ of the DSA, and the data resource strategy of the DSA in the period is

$$x_{ij}^{(\tau+1)} = x_{ij}^{(\tau)} + \lambda d^{(\tau)}, \tag{19}$$

where λ is the data resource strategy adjustment step size and $d^{(\tau)}$ is the gradient of the utility function and its calculation formula is

$$d^{(\tau)} = \left.\frac{\partial U_{Ni}}{\partial x_{ij}}\right|_{x_{ij}=x_{ij}^{(\tau)}}. \tag{20}$$

Because U_{Ni} has been proven to satisfy the characteristics of the concave function, the data resource strategy of the DSA can converge to the Nash equilibrium point through (19) after a plurality of $\Delta\tau$.

The optimal price strategy of FN can be calculated by iteratively adjusting the price and then observing the data resource strategic changes of each DSA to calculate the impact on its utility. The optimal price strategy maximizes the FN's utility function. The price strategy adjustment of the FN can be calculated as

$$p_j^{(t+1)} = p_j^{(t)} + \gamma \frac{\partial U_{Mj}(x(t), p(t))}{\partial p_j(t)}, \tag{21}$$

where $\gamma > 0$ indicates the iteration step size. Similarly, the time interval from time t to time $t+1$ is called an iteration period Δt of the FN. The partial derivative of the price can be calculated by using a small variable ε; the formula is as follows:

$$\frac{\partial U_{Mj}(x(t), p(t))}{\partial p_j(t)} \approx \frac{U_{Mj}\left(\ldots, p_j(t) + \varepsilon, \ldots\right) - U_{Mj}\left(\ldots, p_j(t) - \varepsilon, \ldots\right)}{2\varepsilon}. \tag{22}$$

Before the data resource strategy of the DSA is stable, the price of the FN should be kept constant to obtain the best strategy for the DSA under this price strategy. At this price, the time that the strategy of the DSA achieves stability is called an iteration period Δt of the FN, and one Δt contains multiple $\Delta\tau$. Throughout the framework, the FN as a leader dynamically adjusts pricing based on the demand of the DSA, and when its revenue reaches its maximum, it stops changing the pricing strategy and determines this price as the best price p^*. Under this pricing, the DSA's best data resource strategy x^* is the optimal response to the pricing of the FN. At this point, the leaders and followers of the game reach the Nash equilibrium (x^*, p^*).

The algorithm is divided into two parts, including the price strategy of the FN and the data resource strategy of the DSA:

(1) At each time t, the FN formulates a price strategy based on the marginal effects of formulas (21) and (22).

(2) After the DSA receives the new price strategy and then within each time interval $\Delta\tau$, it adjusts its data resource strategy according to formulas (19) and (20) until the utility reaches the maximum value and the entire DSA reaches the Nash equilibrium.

(3) If the utility of all the FNs reaches the maximum value at this time, the iteration is stopped. Otherwise, at the next time $t+1$, the FN returns to (1) according to the data resource strategy of the DSA to continue iteration.

6. Simulation Experiments

The simulated scene was on a 3,000-meter road. All the cars were running in one direction. The initial DSA service area was 500 meters. In this 3,000-meter area, we allocated 3 FNs and DSAs. We assumed that each car's sensor is in the same location on the car, the rate of data transmission is 50 km/ms, and the delay tolerance of the car is 60 ms. In the iterative algorithm experiment, we only use two service areas covered by different FNs. There are three kinds of DSAs in this area: $\alpha = 0.5$, $\alpha = 0.9$, and $\alpha = 2$. Assume that under the initial conditions, the price strategies of both FNs are 0.1, the value of β is 1, and the initial data resource strategy of the DSA at both FNs is 0.

As the density of vehicles in the area continues to increase, the service area of the DSA becomes correspondingly smaller. In the case where the size of the service area is the same, the density of vehicles in the service area of the DSA with a resource occupancy rate of 0.2 is the largest. When the density of cars is fixed, we find that the DSA with a resource occupancy rate of 0.8 has the smallest service area, that is; when the server is busy, the service area of the DSA is smaller. As shown in Figure 4, both density of vehicles and resource occupancy rate can affect the service area of the DSA; the greater the density of cars, the fewer the resources available to the server and the smaller the service area (see Figure 4).

Figure 5 shows the curve changes of three different utility functions of DSA in the iteration process. The users of curve $\alpha = 2$ are cars, which are the least sensitive to data service delay, so the corresponding price paid by the cars is relatively low. The resource in the FN is preferentially contested by DSA, which is sensitive to delay and applies for more resources, so the effect is relatively low. The curve of $\alpha = 0.5$ is a car and is the most sensitive to data service delays, so the DSA needs to buy more data resources from the FN, and the price strategy of the FN will be more friendly, so the utility of the DSA is getting higher and higher, but as the equilibrium point of the game is reached, the resource of the FN is effectively utilized, FN found a higher price in the Nash equilibrium state, and the utility of the DSA decreases accordingly, so has a maximum value at the beginning. The car in the DSA service area of $\alpha = 0.9$ is moderately sensitive to service delays, and the curve increases gradually at first and

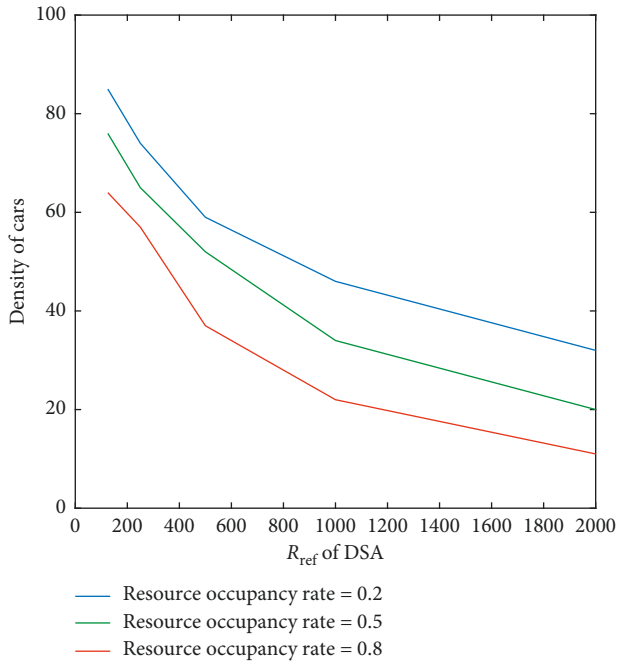

Figure 4: Relationship between the service radius and vehicle density.

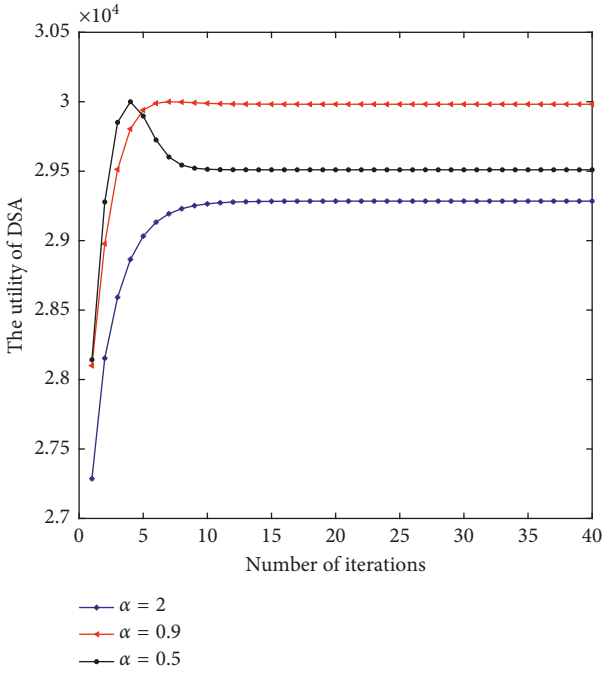

Figure 5: Different user utility changes with the iterative process.

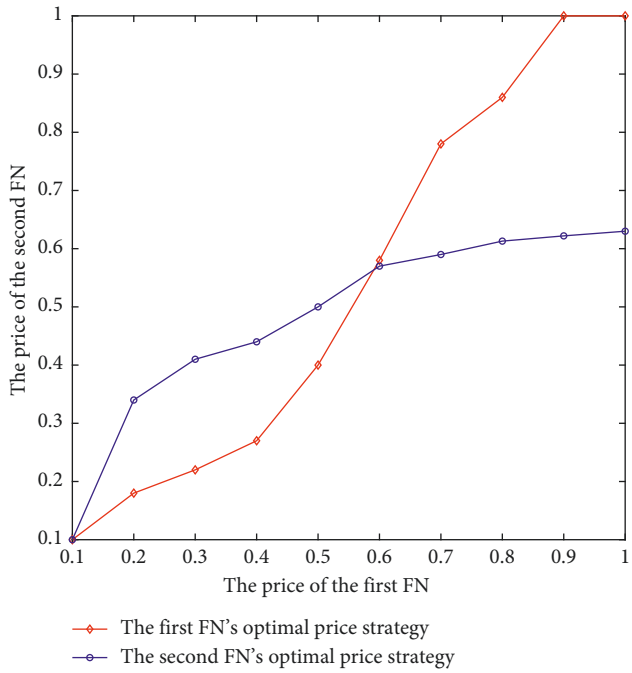

Figure 6: Nash equilibrium of the game between FNs.

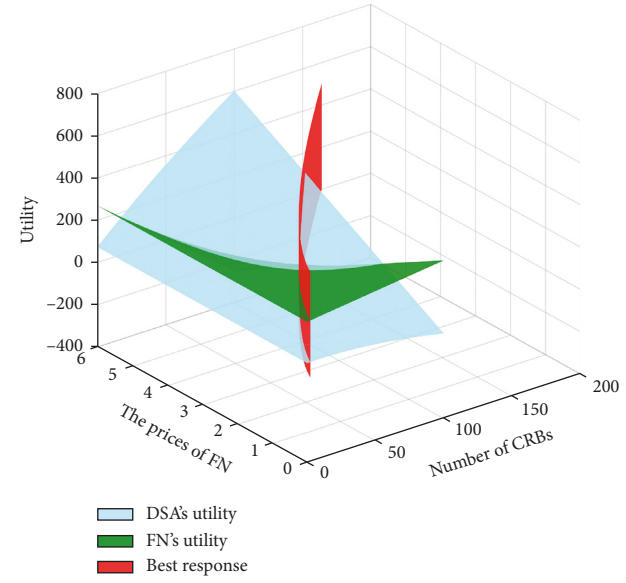

Figure 7: The relationship between price, CRB quantity and utility.

then gradually becomes stable. All three curves tend to stabilize after reaching that equilibrium point (see Figure 5).

We obtain the subgame perfect Nash equilibrium of the Stackelberg game of heterogeneous wireless networks. The two curves in the figure are the optimal price curve of the first FN and the optimal price curve of the second FN. The intersection of the two curves is the Nash equilibrium point p^*. Because the price strategy at this point can satisfy the maximum utility of two leaders at the same time, if any FN unilaterally changes the current price, it will reduce its utility. The intersection of two curves corresponds to the optimal price strategy of two FNs. At this point, both the FN layer and the DSA layer have reached the Nash equilibrium, and the subgame perfect Nash equilibrium is obtained for the Stackelberg game (see Figure 6).

We evaluated the utility of the DSA and the FN and combined the best data service status after each layer of utility. As shown in Figure 7, the utility of the DSA and the FN is affected by the pricing of the FN and the demand for

CRB is determined by the DSA. The FN cannot continue to increase the price of a unit virtual CRB to increase utility because when the price is too high, the DSA will reduce the number of purchases, so the utility of the FN will decrease only when the most suitable price and the best CRB quantity are reached, and the utility of DSA and FN can reach the maximum (see Figure 7).

7. Conclusion

In this paper, we propose a joint optimization framework for multi-FN, multi-DSA, and multicar scenarios for VEC. Under this framework, we first determine the service area of the DSA according to the characteristics of the car movement. Then, we model the Stackelberg game to solve the pricing problem of the FN and the resource purchase problem of the DSA. For each stage of the problem, all participants can achieve balanced or stable results, and no one in this framework can unilaterally change their behavior to achieve higher utility. The simulation results show that all the FN and the DSA can achieve the best effect for themselves and can achieve the high performance of the proposed framework compared. For future work, we can consider the contact between the cars of the lowest layer and whether some data services can be obtained in the neighborhood vehicles. If data service can be provided between vehicles, which vehicle should be selected is determined.

Data Availability

The data used to support the findings of this study are included within the experimental figure in this article.

Conflicts of Interest

The authors declare no conflicts of interest

Authors' Contributions

Ying Zhang proposed the idea of this paper. Guangshun Li and Maoli Wang finished the algorithm and English writing of the paper. Junhua Wu and XiaoFei Sheng completed the experiments.

References

[1] Q. Hu, C. Wu, X. Zhao, X. Chen, and T. Yoshinaga, "Vehicular multi-access edge computing with licensed sub-6 GHz, IEEE 802.11p and mmWave," *IEEE Access*, vol. 6, pp. 1995–2004, 2018.

[2] L. Qi, J. Yu, and Z. Zhou, "An invocation cost optimization method for web services in cloud environment," *Scientific Programming*, vol. 2017, Article ID 4358536, 9 pages, 2017.

[3] L. Feng, J. Yu, F. Zhao, and H. Jiang, "A novel analysis of delay and power consumption for polling schemes in the IoT," *Tsinghua Science and Technology*, vol. 22, no. 4, pp. 368–378, 2017.

[4] 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.

[5] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: a joint optimization approach combining Stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[6] A. V. Dastjerdi and R. Buyya, "Fog computing: helping the Internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[7] Z. Ning, X. Kong, F. Xia, W. Hou, and X. Wang, "Green and sustainable cloud of things: enabling collaborative edge computing," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 72–78, 2019.

[8] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[9] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic load balancing for virtual resource management in datacenters," *IEEE Transactions on Cloud Computing*, vol. 99, p. 1, 2016.

[10] B. Ahlgren, P. Aranda, P. Chemouil et al., "Content, connectivity, and cloud: ingredients for the network of the future," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 62–70, 2011.

[11] M. Yannuzzi, R. A. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Keyingredient sinan iotrecipe: fog computing, cloud computing, and more fog computing," in *Proceedings of the IEEE International Workshop on Computer Aided Modeling & Design of Communication Links & Networks*, Guildford, UK, 2015.

[12] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: a survey of the emerging 5G network edge architecture & orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[13] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and repurchasing for big data processing in multi-clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 266–277, 2016.

[14] R. Hasan, M. Hossain, and R. Khan, "Aura: an incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading," *Future Generation Computer Systems*, vol. 86, pp. 821–835, 2017.

[15] G. Li, S. Xu, J. Wu, and H. Ding, "Resource scheduling based on improved spectral clustering algorithm in edge computing," *Scientific Programming*, vol. 2018, Article ID 6860359, 13 pages, 2018.

[16] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, "Methods of resource scheduling based on optimized fuzzy clustering in fog computing," *Sensors*, vol. 19, no. 9, p. 2122, 2019.

[17] W. Cong, Y. Ying, C. Wang, H. Xi, and C. Zheng, "Virtual bandwidth allocation game in data centers," in *Proceedings of the IEEE International Conference on Information Science & Technology*, Wuhan, China, March 2012.

[18] M. Wu, D. Ye, S. Tang, and Y. Rong, "Collaborative vehicle sensing in bus networks: a Stackelberg game approach," in *Proceedings of the IEEE/CIC International Conference on Communications in China*, Chengdu, China, July 2016.

[19] W. Hao, Y. Zhao, and H. Guan, "On pricing schemes in data center network with game theoretic approach," in *Proceedings of the International Conference on Computer Communication & Networks*, Honolulu, HI, USA, February 2014.

[20] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart

mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.

[21] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Network*, vol. 27, no. 5, pp. 48–55, 2013.

[22] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: a viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[23] A. Alamer, Y. Deng, G. Wei, and X. Lin, "Collaborative security in vehicular cloud computing: a game theoretic view," *IEEE Network*, vol. 32, no. 3, pp. 72–77, 2018.

[24] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 60–66, 2016.

[25] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for RSU clouds in support of the Internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, 2015.

[26] D. Kim, Y. Velasco, W. Wei, R. N. Uma, and S. Lee, "A new comprehensive RSU installation strategy for cost-efficient VANET deployment," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4200–4211, 2016.

[27] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the Internet of vehicles: architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[28] Q. Ranran, *Research on Computational Method of Traffic Accident Vehicle Speed Based on Surveillance Video Image*, Jilin University, Jilin, China, 2017.

[29] N. Pu-yan and Z. Pei-ai, "A note on Stackelberg games," in *Proceedings of the 2008 Chinese Control and Decision Conference*, pp. 1201–1203, Yantai, China, July 2008.

[30] Y. Jiang, S. Chen, and B. Hu, "Stackelberg games-based distributed algorithm of pricing and resource allocation in heterogeneous wireless networks," *Journal on Communications*, vol. 1, pp. 61–68, 2013.