Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 5918128, 13 pages
https://doi.org/10.1155/2022/5918128

Research Article

A Software-Defined Networking Roadside Unit Cloud Resource Management Framework for Vehicle Ad Hoc Networks

Hongming Li[1,2] **Dongxiu Ou**[1,2] **Iftikhar Rasheed**[3] **and Meiting Tu**[1,4]

[1]*The Key Laboratory of Road and Traffic Engineering, Ministry of Education, School of Transportation Engineering, Tongji University, Shanghai 201804, China*
[2]*Key Laboratory of Railway Industry of Proactive Safety and Risk Control, School of Transportation Engineering, Tongji University, Shanghai 201804, China*
[3]*Department of Information and Communication Engineering, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan*
[4]*College of Automatic, University Paris-Saclay, Saint Aubin, Paris 91190, France*

Correspondence should be addressed to Dongxiu Ou; ou.dongxiu@tongji.edu.cn

Received 19 November 2021; Accepted 11 January 2022; Published 8 February 2022

Academic Editor: Alain Lambert

Roadside unit (RSU) cloud and its vehicle-to-infrastructure (V2I) connectivity can enable various security, entertainment, and shared mobility applications for vehicles in intelligent transportation systems (ITS) through wireless communications. In this article, the deep programmability of software-defined networking (SDN) is employed to dynamically reconfigure network hosting services and their data forwarding information for effectively meeting the basic shared mobility applications' needs in vehicle ad hoc networks (VANETs). Multipath is also enabled to forward data flow for balancing network links utilization rate and SDN is thus utilized to achieve the minimum cloud delay with the least number of hosts, which can be summarized as a mixed-integer linear programming (MILP) problem. The joint optimization (JO) algorithm is proposed and in contrast to the two single-objective algorithms which are the delay optimization (DO) algorithm and host optimization (HO) algorithm, respectively. Results show that, for the single-threading instance, the JO and DO algorithms are the same in essence. For the multithreading instance, the JO algorithm generally outperforms the two single-objective optimization algorithms, respectively, under given demands. Furthermore, results also demonstrate that the services should be deployed globally in a distributed manner rather than in the centralized manner for achieving the minimized cloud delay in designing an RSU cloud.

1. Introduction

Transportation is leading an important service industry in the national economy while traffic congestion and safety are the major problem faced by almost all core cities in the world [1]. The traditional method of controlling traffic flow based on traffic lights and stop signs cannot dynamically adjust the interval time in real-time according to the actual traffic flows. Variable speed limit is a major ITS technology for controlling freeway mainline traffic, which have been increasingly used to improve traffic safety and operations efficiency of freeway traffic management [2]. What is more, electric vehicles and its related applications are the main trend in recent years, and its configuration optimization to prolong driving mileage has become an essential issue in affecting the future prospect of electric vehicles [3]. Though there are some shared mobility applications, such as ride splitting, enabling riders with similar routes to share a ride sourcing trip, which is a promising transportation technology to reduce traffic congestions and air pollution, the traffic management can be enhanced through effective communications among the vehicles [4, 5]. However, the pure vehicle-to-vehicle (v2v) communications are unstable, due to the occlusion of towering buildings on the roadside or the other obstacles resisting the communication link, in which case the links are likely to be unstable and decrease the communication quality. Besides, the network topologies are dynamic, because of the high-speed movements of the vehicles. Therefore, the established communication links among vehicles may be disconnected, which affects the success rate of data transmission. In V2V, the communications among the vehicles are not stable enough and the quality of communication is determined by many uncontrollable factors, such as vehicle speeds, geographical locations, and weather conditions. Considering the problems that existed in the V2V communications, the RSUs are utilized for assisting the efficient communications to form the VANETs environment.

The RSUs play a vital role in the VANETs system and are usually deployed on both sides or the middle of the road, which can be equivalent to an AP (access point) in a wireless network. Vehicles driving on the roads can use RSUs to realize interconnection and intercommunication. At the same time, the existence of RSUs makes the network more stable and reliable when uploading and downloading data, and the bandwidth will increase accordingly. To improve the network performance in the VANETs, Yang et al. propose a connectivity-oriented maximum coverage RSU (CMCS) deployment scheme which aims to maximize the performance of V2I communication in urban environments [6]. Another role of the RSUs is to provide computing services and serve as a temporary storage point for data. Some data content such as video data with high throughput rates from vehicles and road condition information that are commonly requested can be temporarily stored in RSUs, without repeated requests to the Internet. VANETs play a vital role in assisting driving, traffic accident warning, traffic management, and Internet services. To better adjust the traffic congestion situation, intelligent transportation system (ITS) applications would be of great importance with the help of VANETs [7]. Under this circumstance, smart cars can automatically adjust the interval between traffic lights through interaction with the local vehicle-road collaboration system under VANETs environment, dynamically optimize the traffic flow at road intersections, improve vehicle traffic efficiency, and alleviate congestion.

The main communication ways in VANETs include the communications among vehicles named V2V (vehicle-to-vehicle) and the communications between the vehicles and RSUs called V2I (vehicle-to-infrastructure), where the typical features of V2V are described above. As far as V2I communication is concerned, due to the fixed geographic location of the RSUs, the relatively stable network, and relatively few interference factors, V2I has been studied in-depth. At present, there are still many problems in V2I communication, such as access selection, data distribution, and frequent switching of RSUs, which restrict the further development of VANETs and cannot provide users with more real-time and high-quality communication services. In terms of RSUs service, due to the high dynamics of VANETs and the limited coverage areas of RSUs, multiple vehicles enter the areas covered by RSUs at the same time. Delay-sensitive safety information would probably result in not being delivered in time, and nonsafety application entertainment information might occupy most of it. It is related to whether the vehicles can get timely processing information and satisfactory services. It will also affect the operating status of the RSUs in the entire region, including throughput, bandwidth utilization, and load balance. Network congestion sometimes occurs in some RSUs, while other RSUs have idle bandwidth and waste. The unbalanced load of RSUs will directly lead to the degradation of network performance [8]. Therefore, how the vehicles choose the appropriate RSUs to get served will be of great importance. In the RSU cloud where there are multiple RSUs, a more scientific algorithm is needed to obtain the best solution for vehicles to get served so that the performance of all aspects of the network is maintained stable. In the same way, in the current VANET data distribution system, due to the limitation of the coverage of RSU and the solidification of the Internet of vehicles architecture and communication methods, the service instances of the vehicles need to be transferred or copied to other RSUs when transmitting large-volume data.

Various methods have been studied with different ways to achieve efficient communications among the vehicles in the ITS systems under VANETs environment. As a matter of fact, the RSU clouds are proposed to help build the VANETs environment, for example, to increase the number of communication links for sending data packets. When the service of the current vehicles cannot be met or need to be transferred, deleted, or copied to other RSUs, how to manage the computing and communication resources in the RSU cloud is an essential problem. Therefore, to overcome this problem, the SDN-based approach with an RSU-based resources management framework for VANETs is proposed. The problems are firstly focused on how to manage the RSUs' characters in the process of setting data dissemination policies and then optimize the RSUs' resources deployment with the dynamic existing communication demands. In this case, a mixed-integer linear programming (MILP) model is developed to optimize the allocation of the RSUs' resources in the designed SDN RSU cloud architecture while minimizing the communication delays and number of working hosts. Furthermore, the RSUs' resources redundancy is also considered upon the basis of ensuring the requirements of communication delay and working hosts' number.

For the rest part of the paper, the related work is firstly reviewed and then the proposed SDN-based RSU cloud framework is introduced. Next, the SDN-based RSU cloud resource management problem is formulated into the mixed-integer linear programming (MILP) problem with some restrictions, which is solved by building a simulation framework using Python that is mixed programmed with Gurobi. Meanwhile, the JO algorithm is compared with the single-objective optimization algorithms which are HO and DO algorithms, respectively. Finally, the paper is concluded, and the future work is summarized.

2. Literature Review

To ensure the efficient communication in the VANETs, constructing the cloud of OBUs and (or) RSUs and (or) the hybrid cloud are the typical ways for achieving the generality of the entire network. In this section, lots of existing work on the various clouds in the VANETs shall be briefly reviewed.

On the one hand, a cloud of on-board units (OBUs) are playing vital roles in the VANETs, where some RSUs still act as the gateways to the traditional communication networks [9]. Behbehani et al. [10] introduce a vehicular cloud architecture supported by fixed edge computing nodes and the central cloud; mixed-integer linear programming (MLP) model is developed to optimize the allocation of the computing demands in the distributed architecture while minimizing power consumption. Nodes that are consisted of "SmartCloud" vehicles in the minimum connected dominating set form a virtual backbone in the VANETs [11], where a "SmartCloud" vehicle sends periodic beacon messages to the nearest RSU. For the above method, the common communication evaluation metrics such as the transmission range, processing power, and delay reduction are greatly improved. Meanwhile, the methods mentioned above mainly devote their efforts into the utilization of the vehicles' OBUs, rather than taking advantages of the RSUs.

On the other side, Wang et al. [12] construct a scene where information centres (or base stations) are distributed along the road so that the information centres can broadcast messages in time, aiming to work out the information dissemination problem in a joint vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication system. Though it is possible to interact between the vehicles and roads in the local area, the decision-making is based on the individuals or the local groups, they do not consider the problem from the global situation in a road network, which cannot make full use of the RSUs' extra communication and computing resources and it is impossible to achieve the overall situation from the perspective of safety or efficiency optimization. Meanwhile, cloud computing is a new and promising paradigm delivering IT services as computing utilities, which is designed to provide services to external users, providers for sharing their resources, and capabilities [13]. For a cloud which is composed of vehicles, RSUs, and passengers, the traffic flows are usually distributed in an unbalanced way in the region, and it remains dynamic as time goes by, the types of the ITS applications and the bandwidth demands of the vehicles are different for the various cloudlets of the entire network. Therefore, it is time to make full use of the RSUs' extra communication and computing resources with less demands, letting the RSUs with larger demands alleviated in serving the cars. Tang et al. have proposed an RSU empowered vehicular network that consists of three hierarchical layers-vehicular cloud, RSU-enabled cloudlet, and central cloud, respectively, where RSUs are enhanced with edge servers so that the entire system can intelligently respond to the real-time resource requests [14]. Zheng et al. employ the knapsack problem to formulate the problem of cooperative scheduling and radio resource management in vehicular networks, but their research only orients to the radio resources for nonreal-time services where they assume the problem in the 2-hop vehicular network, greatly restricting the application scenario [15]. Their proposed methods are not efficient enough considering the high mobility of vehicles in VANETs with strict delay requirements. In fact, VANET evolves with two emerging paradigms: vehicle cloud computing and information centric networks, where vehicle cloud computing brings the mobile cloud model to vehicle networks and information centric networks change the notion of data routing and dissemination [16].

Besides, to date, there is no set architecture for cloud computing from academic point. Olariu et al. envisioned the paradigm shift from conventional VANET to vehicular cloud computing by merging VANET with cloud computing [17], whereas they did not carry out their thoughts in a specific scenario. With the development of computer networks, McKeown et al. have proposed OpenFlow which is the de facto standard of software-defined networks, where it can provide an excellent opportunity to improve network performance through flexible centralized control, decoupling control plane (network control functions) from the data plane (forwarding functions) [18]. The control plane communicates with the data plane using OpenFlow protocol, where the controller enables the ITS operators to set data dissemination policies and integrate services. Salahuddin et al. have proposed a unique SDN Cloud RSU architecture and its SDN advances from the previous in two parts: firstly, it consists of traditional RSUs and small data centres; secondly, it can be dynamically reconfigured to meet changes in service demands ensuring QoS [19]. However, they do not take the RSUs' resources redundancy into consideration in case that the networks communications demands are too fluctuant to meet. In this research, a mixed linear integer (MILP) programming model is developed to optimize the RSU resource allocation in the designed SDN RSU cloud architecture. While minimizing communication delays and the number of hosts in the cloud, the redundancy of RSU resources is also considered.

3. Materials and Methods

Since that SDN was proposed as a new-generation network architecture, its unique concept of separation for data and control brings us convenience and flexibility in designing a cloud. The SDN architecture based on the OpenFlow protocol is divided into three planes: application plane, control plane, and infrastructure plane. Control plane mainly includes OpenFlow controller and network operation system. The OpenFlow controller encapsulates the northbound interface driver that connects to the required plane; the southbound interface driver connects to the switch, ensuring the internal intelligent logic of SDN control. The control plane realizes data transmission with OpenFlow switches via the southbound interface, obtaining the status information of the underlying infrastructure and performing reasonable scheduling of the underlying network resources. Obviously, the controller is the key to the centralized control function of the entire SDN network. To achieve the best performance of the RSU cloud infrastructure, it is also necessary to implement multipaths in an SDN that supports OpenFlow. In the proposed SDN RSU cloud framework, the RSUs are composed of the physical servers which can store and compute data and switches that can exchange nonsafety apps information from the vehicles on the road sections and nearby RSUs. The physical machine of the RSU enables virtualization so that the virtual machines can run multiple replications to host services on a single physical machine. Meanwhile, the services can be deleted, transferred, or copied to other RSUs when the vehicles are travelling to the next RSU. The switches on the RSUs are all OpenFlow switches which are the de facto standards of SDN, performing as the corresponding functions of the data forwarding layer. However, though there is no stringent QoS requirement for ITS apps, it is highly desirable and usually crucial for its users [20]. The proposed SDN RSU cloud framework is shown in Figure 1.

Figure 1 depicts the SDN RSU cloud framework and there are n $(0 \leq n < N)$ RSUs which are connected to each other through a high-speed backbone. The development of the vehicle-road collaborative technology via cloud computing aims to achieve vehicle-road information sharing system based on advanced communication technologies by deploying roadside intelligent sensing computing equipment (collectively referred to as roadside infrastructure) and complementing the car-side through roadside perception. The RSUs are deployed in the centre of the roads or intersections for maximizing the coverage area of the vehicles to be served. For the entire region, since there are more vehicles on the arterial roads and less vehicles on the secondary roads, the demands are unbalanced distributed. The unevenly distributed network demands would greatly fluctuate as the traffic density changes. As the density of traffic flows is changing with time, network demands are varying to a large extent. Due to the varying network demands, it is necessary to dynamically reconfigure the SDN RSU cloud for optimizing the communication efficiency.

There are three RSU function characters which are the OpenFlow controllers, RSU cloud resource managers (CRMs), and normal RSUs. OpenFlow controllers are responsible for formulating and managing the forwarding strategy of the flow table which can update the OpenFlow switch flow rules via the control plane. Meanwhile, the CRMs would disseminate information regarding service hosting, service migration, data flow changes, and virtual machines (VMs) instantiating and (or) eliminating via the data plane; the designed characters are simplified compared with the research by Salahuddin et al. [21] where there are three characters. It is worth mentioning that the RSU cloud resource managers communicate with the OpenFlow controllers via the data plane.

The top layer of the architecture in Figure 1 is the Internet, which contains a large quantity of Internet servers to provide data upload and download services. The RSUs in the cloud not only transmit information of ITS services but also serve as the different characters, respectively, ordered by the controller information that the network manager has given. The fixed SDN RSU cloud upper layer connections in Figure 1 are all wired broadband connections, including the connections between normal character RSUs and other normal character RSUs, the normal character RSUs and OpenFlow controllers, normal character RSUs and CRMs, and OpenFlow controllers and CRMs, ensuring the reliability of the data transmission. The communications between mobile vehicles and the RSUs adopt IEEE 802.11p wireless communication. There are mostly RSUs with normal characters which are given nor the function of OpenFlow controllers neither cloud resource managers as shown in Figure 1.

4. SDN RSU Cloud Resource Management

Cloud resource management pays attention on the optimal allocation of computing and network resources while meeting the fluctuating demands required by the ITS apps. Normally, optimizing the placement of virtual machines on physical host machines is determined by various capacity planning tools such as VMware Capacity Planner, IBM WebSphere CloudBurst, Micro Focus, and OneIQ; these cloud resource management tools seek to consolidate physical memory, VMs for CPU, and power consumption savings, etc., yet without considering consumption of network resources [22]. Consequently, some RSUs will inevitably cost lots of network resources, while others are almost idle, causing the unbalanced load of the entire network, which would directly lead to the degradation of network performance. Defining the configuration as the number of service hosts and effective data forwarding rules, the mission is to find the best configuration in order that the whole SDN RSU cloud will achieve the minimum delay and minimum number of hosts when the network demands change.

4.1. Problem Formulation.

4.1. Problem Formulation. Considering a commonly seen traffic network which is fixed with a bunch of RSUs facing the fluctuating communication demands, the mathematical problem is then abstracted as follows. Assume that a network graph $G = (Vertices, Edges)$ is given with fixed number of RSUs, whose $vertices\, and\, edges$ are representing the RSUs and the wired connections among the RSUs, respectively.

From what has been shown in Figure 2, given a set of traffic information services S, a set of average demands $D = \{D_{t_0}, D_{t_1}, D_{t_2}, \ldots, D_{t_n}\}$ that change over time periods $T = \{t_0, t_1, t_2, \ldots, t_n\}$. The vehicles are constantly sending their demands at each RSU and requesting for services from the RSU cloud. There is an initial network configuration Z_{t_0} during period t_0. For any edge, there is a bandwidth capacity C_e. During the time period, it is assumed that there is a demand $b_{n,k}$ at RSU n for service k with the average demand D_{t_i}. The mathematical model is designed to compute the optimal network configuration while minimizing the host numbers and RSU cloud delay during period t_i, under the premise of ensuring a certain amount of service resources redundancy. In this case, when some over-burdened RSUs send part of demands to other RSUs with less demands, the demands in the cloud are all met with a load-balanced network.

Defining the $delay$ as the sum of the delays in all edges that a single routing path traverses from RSU m to n, which includes processing time T_P, queuing time T_q, transmission time T_{tr}, and propagation time T_{pg}, processing time T_P is a constant, which is usually 10 μs. For the queuing process of a data packet, the Poisson distribution combined with the traffic background is used for modelling. For the Poisson

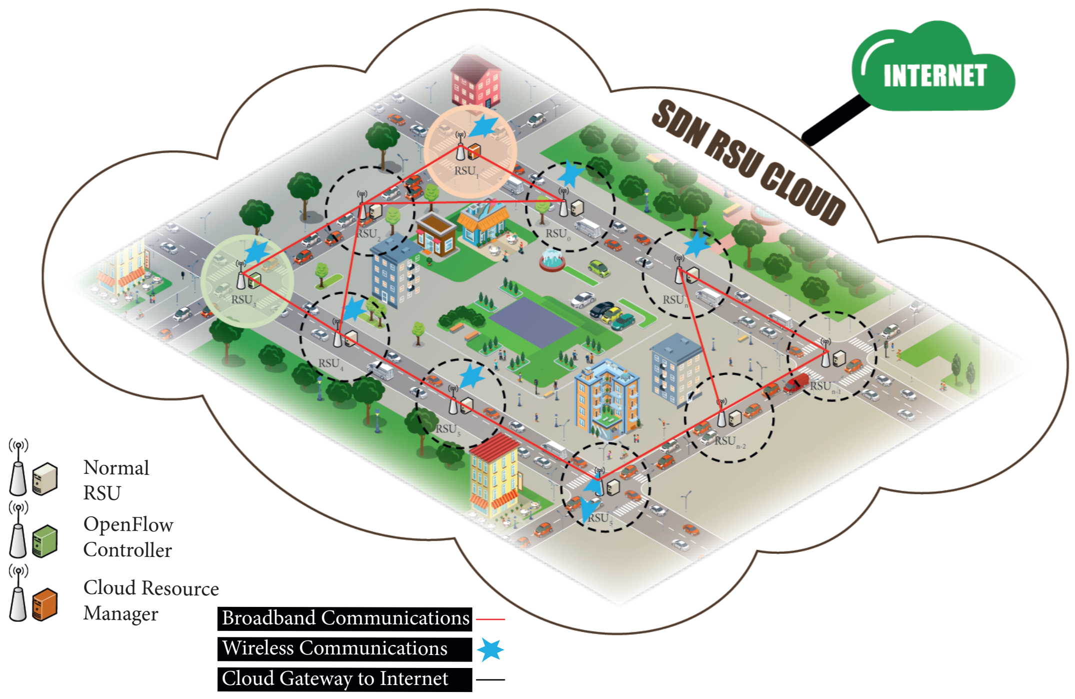


Figure 1: System architecture of the proposed SDN RSU cloud.

process with the packet interarrival times λ and packet processing times μ, whose mean and standard deviation are λ_a, σ_a and λ_p, σ_p, respectively, the coefficients $Cof_a = \sigma_a/\lambda_a$, $Cof_p = \sigma_p/\lambda_p$ are defined by the Kingsman formula to describe queuing delay T_q[23]:

$$T_q = \frac{Cof_a + Cof_p}{2} \cdot \frac{\lambda/\mu}{\mu - \lambda}. \tag{1}$$

According to the basic knowledge of physics, the transmission speed of signal in the wire equals about 2/3 light speed. Propagation time T_{pg} is determined by the size of packet divided by the capacity C_e:

$$T_{tr} = \frac{\text{length}}{(2/3) \cdot \text{light speed}}, \tag{2}$$

$$T_{pg} = \frac{\text{size of packet}}{C_e}. \tag{3}$$

As for the convenience of describing the parameters in the model, the notations of the input parameters are listed in Table 1, and output parameters and constants are depicted in Table 2, respectively.

As what can be seen in Table 1, the essential variables are all displayed with the detailed explanations and there are several binary variables which are of great significance in determining the output results. In the output parameters, there are also several variables which are corresponding to the ones in Table 1, which are listed in Table 2.

Table 2 lists the output parameters and constants, parts of which are related to some variables in Table 1. Since all the parameters needed for this paper are depicted in detail above, it is time to design a joint optimization model to find the best configuration that can satisfy the delay requirements and minimize the number of hosts simultaneously. For the implementation of the SDN RSU cloud problem, two objective functions are involved: the minimization of the delay problem, which is designed to be at a given bandwidth to optimize the delay between the RSUs and users, and the minimization of service hosts number problem, which aims to deploy the minimum number of service hosts to maximize the entire network's service capabilities. What is more, all the things would be done under the premise of ensuring a certain amount of service resources redundancy so that the network instability caused by sudden service demand increase would be prevented. The object function of the problem is formulated in equation (4) and the formula X rounds the number X to the next largest integer.

$$\min \begin{cases} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{j=0}^{\Omega_{m,n}-1} d_{m,n,j}, \\ \sum_{m=0}^{N-1} \sum_{k=0}^{S-1} \lceil h_{m,k} \cdot R \rceil. \end{cases} \tag{4}$$

In the process of solving this multiobjective function, due to the reason that the priority of the two goals is hard to be determined, two strategies are tried to control the priority for the object function. That is, the strategy that the delay takes priority of the host numbers, its worked-out results are compared to the results of the strategy that the host numbers take priority of the delay. After trying many times, finally, it is found that the priority of the two goals has almost no effect on the working-out results, and the priority of the two goals is set to equal with each other. The goal of achieving the minimum service hosts number is contradictory to the minimization of delay. By transforming the proposed multiobjective problem into a single-objective optimization problem, linear scaling technology is used to solve the proposed multiobjective problem [24]. The delay

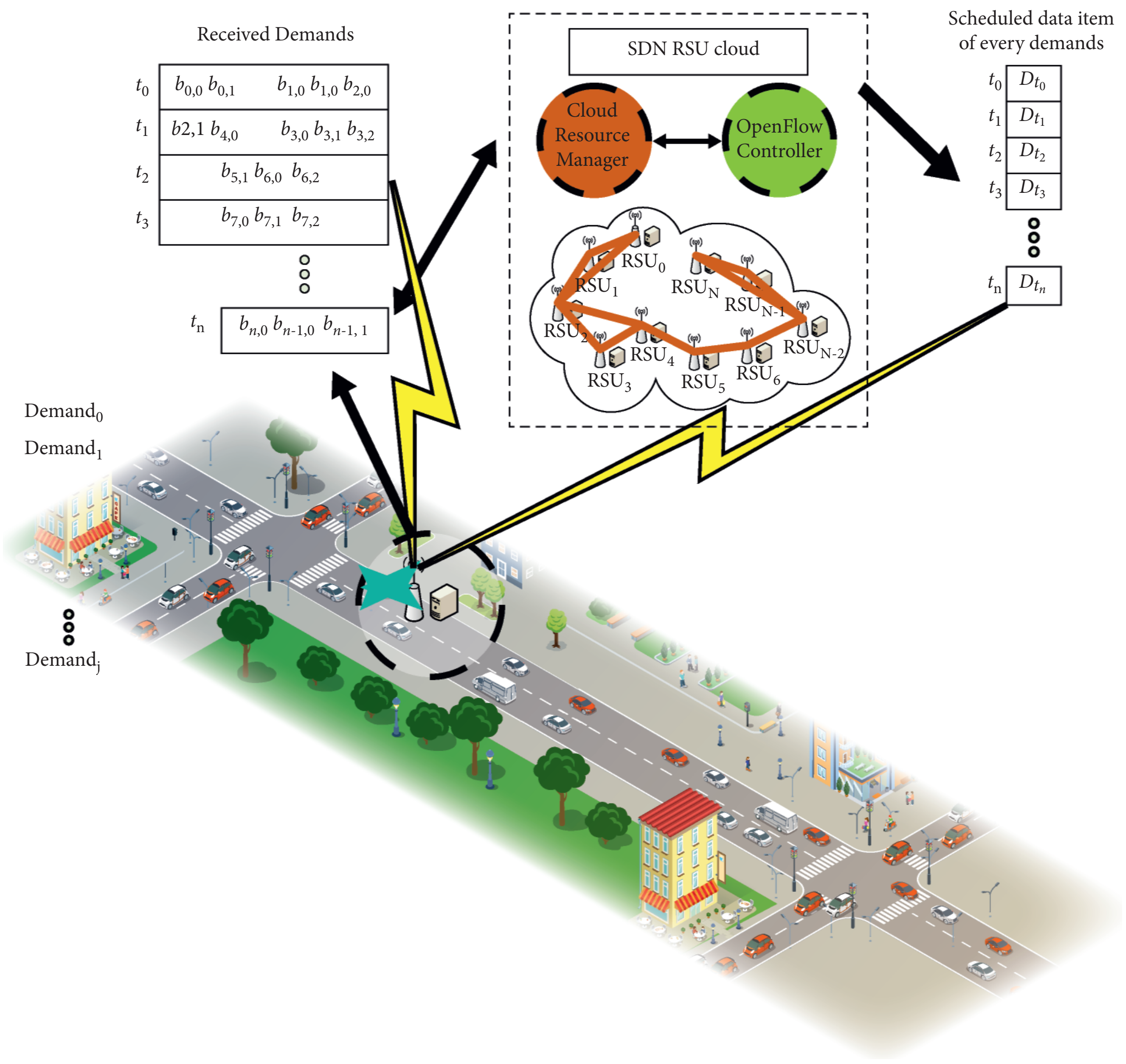

Received Demands
t_0 $b_{0,0}$ $b_{0,1}$ $b_{1,0}$ $b_{1,0}$ $b_{2,0}$
t_1 b2,1 $b_{4,0}$ $b_{3,0}$ $b_{3,1}$ $b_{3,2}$
t_2 $b_{5,1}$ $b_{6,0}$ $b_{6,2}$
t_3 $b_{7,0}$ $b_{7,1}$ $b_{7,2}$
t_n $b_{n,0}$ $b_{n-1,0}$ $b_{n-1,1}$
SDN RSU cloud
Cloud Resource Manager
OpenFlow Controller
RSU$_0$
RSU$_1$
RSU$_2$
RSU$_3$
RSU$_4$
RSU$_5$
RSU$_6$
RSU$_N$
RSU$_{N-1}$
RSU$_{N-2}$
Scheduled data item of every demands
t_0 Dt_0
t_1 Dt_1
t_2 Dt_2
t_3 Dt_3
t_n Dt_n
Demand$_0$
Demand$_1$
Demand$_j$

Figure 2: The mechanism of the proposed SDN RSU cloud.

requirements are firstly ensured, and then the number of hosts is minimized. Besides, the constraints of the goal (4) are listed from (5) to (22) as follows:

$$B \cdot \chi_{m,n,j} \geq h_{m,k} - \eta_{m,k}, \ \forall 0 \leq k < S, 0 \leq m < N, \quad (5)$$

$$h_{m,k} - \eta_{m,k} + \left(1 - \chi_{m,n,j}\right) \cdot B \geq 0, \ \forall 0 \leq k < S, 0 \leq m < N. \quad (6)$$

The constraints (5) and (6) restrict the definition of the variable $\chi_{m,n,j}$, where if $h_{m,k} - \eta_{m,k} > 0$, $\chi_{m,n,j}$ equals 1, otherwise 0.

$$\sum_{m,j} \beta^k_{m,n,j} \geq b_{n,k}, \ \forall 0 \leq k < S, 0 \leq m, n < N, \quad (7)$$

$$B \cdot h_{m,k} \cdot \zeta_{n,k} \geq \sum_{j} \beta^k_{m,n,j}, \ \forall 0 \leq k < S, 0 \leq m, n < N, \quad (8)$$

$$h_{m,k} \cdot \zeta_{n,k} \leq \sum_{j} \beta^k_{m,n,j}, \ \forall 0 \leq k < S, 0 \leq m, n < N, \quad (9)$$

$$\beta^k_{m,n,j} \leq B \cdot p^k_{m,n,j}, \ \forall 0 \leq k < S, 0 \leq m, n < N, 0 \leq j < \Omega_{m,n}, \quad (10)$$

$$\beta^k_{m,n,j} \geq p^k_{m,n,j}, \ \forall 0 \leq k < S, 0 \leq m, n < N, 0 \leq j < \Omega_{m,n}. \quad (11)$$

The general communication processes can be concluded as if demands exist, there must be loads. As a matter of fact, the RSU servers must meet or exceed the demand of the vehicles, listed as (7). Constraints (8) and (9) ensure that the routes carrying the network loads are only between the RSU servers and the users. To step further, (10) and (11) ensure that there are enough loads for carrying the rule of the control plane which is the natural cost of the system itself.

$$l_e = \sum_{m,n,j,k} \alpha^e_{m,n,j} \cdot \beta^k_{m,n,j}, \ \forall 0 \leq e < |\text{Edges}|, \quad (12)$$

$$l_e = \xi \cdot \tau_e, \ \forall 0 \leq e < |\text{Edges}|, \quad (13)$$

Table 1: Input parameters.

Input	Description
S	Number of services.
N	Number of RSUs.
$b_{n,k}$	The demand at RSU n for service k, $\forall 0 \le n < N, 0 \le k < S$.
$\zeta_{n,k}$	$\zeta_{n,k} = 1$ if a demand happens on RSU n for service k; otherwise, $\zeta_{n,k} = 0$, $\forall 0 \le n < N, 0 \le k < S$.
$\Omega_{m,n}$	Number of paths from RSU m to RSU n, $\forall 0 \le m, n < N$.
$\alpha^{e}_{m,n,j}$	1, if RSU m to RSU n uses path j with edge e; 0, otherwise. $\forall 0 \le m, n < N$, $0 \le j < \Omega_{m,n}$, $0 \le e < \lvert Edges\rvert$.
C_e	Bandwidth capacity of edge e, 100 Mbps, is given here, $0 \le e < \lvert Edges\rvert$.
ϖ_m	RSU m's threshold on number of service hosts, $\forall 0 \le m < N$.
ϖ_k	Service k's threshold on number of hosts, $\forall 0 \le k < S$.
φ_k	Delay threshold for service k, $\forall 0 \le k < S$.
ξ	For controlling the granularity of the lookup table, $\forall 0 \le \xi < C_e$.
$\theta_{i,e}$	Load i's delay on edge e, $\forall 0 \le i < C_e$, $0 \le e < \lvert Edges\rvert$.
$\eta_{m,k}$	1, if service k was hosted on RSU m in the past; 0, otherwise. $\forall 0 \le k < S$, $0 \le m < N$.
$\pi^{k}_{m,n,j}$	1, if a control plane existed for service k on path j between RSUs m and n; 0 otherwise. $\forall\ 0 \le k < S$, $0 \le j < \Omega_{m,n}$, $0 \le m, n < N$.

Table 2: Output parameters and constants.

Output and constants	Description
$h_{m,k}$	1, if service k is hosted on RSU m; 0, otherwise. $\forall\ 0 \le k < S, 0 \le m < N$.
$\beta^{k}_{m,n,j}$	The load for service k on path j from RSU m to RSU n, $\forall 0 \le k < S, 0 \le m, n < N$, $0 \le j < \Omega_{m,n}$.
$p^{k}_{m,n,j}$	1, if a control plane now exists for service k on path j between RSUs m and n; 0, otherwise. $\forall 0 \le k < S$, $0 \le j < \Omega_{m,n}$, $0 \le m, n < N$.
l_e	The load on edge e, $0 \le e < \lvert Edges\rvert$.
d_e	The delay on edge $e, 0 \le e < \lvert Edges\rvert$.
τ_e	The load on edge e is mapped to a bunch of ξ.
$F_{e,i}$	1, if the load on edge e equals to $i \cdot \xi$; 0, otherwise. $\forall 0 \le i < C_e, 0 \le e < \lvert Edges\rvert$.
$d_{m,n,j}$	The delay on path j from RSU m to n, $0 \le j < \Omega_{m,n}, 0 \le m, n < N$.
$\chi_{m,n,j}$	1, if $h_{m,k} - \eta_{m,k} > 0$; 0, otherwise. $\forall 0 \le m, n < N, 0 \le k < S$.
B	A large constant.
R	A constant, usually greater than or equal to 1. The parameter size can be adjusted according to different scenarios to ensure a certain amount of service resource redundancy.

$$\tau_e = \sum_{i=0}^{(C_e-1)/\xi} i \cdot F_{e,i},\ \forall 0 \le e < \lvert \text{Edges} \rvert, \tag{14}$$

$$\sum_{i=0}^{(C_e-1)/\xi} \mathcal{F}_{e,i} = 1,\ \forall 0 \le e < \lvert \text{Edges} \rvert, \tag{15}$$

$$d_e = \sum_{i=0}^{(C_e-1)/\xi} \mathcal{F}_{e,i} \cdot \theta_{i,e},\ \forall 0 \le e < \lvert \text{Edges} \rvert. \tag{16}$$

Equation (12) describes the loads on the *edges* of the flow between RSU m and RSU n, which contains the generating and forwarding traffic on the routes. The delay on the edge matches with the load on it, where the index is used to find it in the lookup table (LUT). Equation (13) represents that the load on an edge e is mapped to a bunch of ξ. Constraint (14) ensures that the traffic load on an edge cannot exceed its capacity, where (15) and (16) state that there is only one specific value for the traffic load on edge e and the delay corresponding to it.

$$d_{m,n,j} = \sum_{e=0}^{\lvert \text{Edges} \rvert - 1} \alpha^{e}_{m,n,j} \cdot d_e,\ \forall 0 \le j < \Omega_{m,n}, 0 \le m, n, m \ne n < N, \tag{17}$$

$$d_{m,n,j} \cdot h_{m,k} \cdot \zeta_{n,k} \le \varphi_k,\ \forall 0 \le j < \Omega_{m,n}, \\ 0 \le m, n, m \ne n < N, 0 \le k < S, \tag{18}$$

$$d_{m,n,j} \cdot h_{m,k} \le d_{m,n,j},\ \forall 0 \le j < \Omega_{m,n}, 0 \le m, n, m \ne n < N, 0 \le k < S. \tag{19}$$

Moreover, the delay of the specific path between the RSU m and n equals the summation of all the delays on its edges, which is shown in (17). Constraint (18) represents the service k's delay threshold, while constraint (19) describes whether a routing path is being used or not.

$$\sum_{m=0}^{N-1} h_{m,k} \ge 1,\ \forall 0 \le m < N, 0 \le k < S, \tag{20}$$

$$\sum_{m=0}^{N-1} h_{m,k} \le \varpi_k,\ \forall 0 \le m < N, 0 \le k < S. \tag{21}$$

On the other side, constraint (20) ensures that there is at least one host working on the RSU m. What is more, constraint (21) represents service k's threshold on number of service hosts. Salahuddin et al. [19] assume that all RSUs have unlimited resources to meet any service demand for simplicity. However, it would be hard and nearly impossible to make the RSUs working without resource restriction.

To further figure out this problem, it is necessary to take the number of working hosts for a single RSU into consideration, which is usually limited.

$$\sum_{k=0}^{S-1} h_{m,k} \le \varpi_m, \ \forall 0 \le m < N. \quad (22)$$

The maximum number of hosts for a single RSU is limited, which is reflected in constraint (22). Besides, formula (22) is used to present the resource constraints in sharp contrast to the work of Salahuddin et al. [19], where they assume that the RSU's resource is infinite to hold the services.

4.2. Hardness Analysis. In this section, the complexity of the proposed model is analysed via testing the complexities of the two terms of it. For the minimization of the delay problem in the implementation of SDN RSU cloud, all OpenFlow switches utilized to route the traffic loads in the network are multipath enabled.

Theorem 1. *The minimization of the delay problem is NP-hard.*

Proof 1. The minimization of the delay problem can be converted into the minimum multicommodity flow (MCF) problem [25]. For an example of the MCF, where a set of nodes Nd exists, which contains k elements $\{Nd_0, Nd_1, Nd_2, \ldots, Nd_k\}$ and an edge set $edges$ with m elements $\{e_0, e_1, e_2, \ldots, e_m\}$, there is a total of K traffic demand, and each traffic demand contains a source node m and a destination node n. A route exists from source m to the destination n whichever paths it chooses when multipath is enabled. The delay on each edge in the network topology is d_e. A graph is therefore formed by all the nodes and edges, where each node Nd in the network is considered as a point in the graph. The delay of a path corresponds to the sum of the delay on every edge it traverses.

It is easy to prove that there is a minimum path delay if and only if there is an optimal solution to the MCF problem via mathematic construction. The construction can be done in polynomial time and the MCF problem is an NP problem [25]. Therefore, minimization of the delay problem is also an NP-hard problem.

To ensure the best reachability and reliability of the entire network, it is best to enable all the RSUs to work on with the threshold number of hosts simultaneously. However, it is wasteful and not acceptable for the users nor for the providers from a pure cost of money perspective, since high operating cost would cost a lot of money. The minimization of service hosts number problem, which is aiming at deploying the minimum number of service hosts without exceeding the threshold while serving all flows in the network. □

Theorem 2. *The minimization of service hosts number problem is NP-hard.*

Proof 2. The minimization of service hosts number problem can be converted into the minimum weighted vertex coverage (MWVC) problem. Here it shows an example of the MWVC problem, containing a set of nodes Nd, which has k elements $\{Nd_0, Nd_1, Nd_2, \ldots, Nd_k\}$ and an edge set $edges$ with m elements $\{e_0, e_1, e_2, \ldots, e_m\}$. These nodes have corresponding weights $\{w_0, w_1, w_2, \ldots, w_k\}$. All nodes and edges form a graph. Here, it treats the network flow set F as a set of edges $edges$ and takes each vehicle or RSU in the network as the node Nd_i in the graph. Therefore, the flow corresponding to the edges passes from the origin Nd_o connected to the destination Nd_d, and the hosts number of each traffic flow corresponds to the weight w_d of destination Nd_d.

Through mathematic construction, it is easy to prove that only and only if there is an optimal solution to the MWVC problem, the SDN RSU cloud can serve all flows with the minimum number of hosts service. The construction can be done in polynomial time, and the minimum weighted vertex cover problem is NP-hard to solve [25]. Hence, the minimization of service hosts number problem is also an NP-hard problem.

Furthermore, $d_{m,n,j}$ is a continuous variable that is larger than 0, while $h_{m,k}$ is an integer variable. Therefore, the proposed model is a mixed-integer linear programming problem (MILP). The MILP problem is solved with the Python-programmed simulation framework which is mixed programmed with Gurobi. All the work done above for minimizing the delay and number of hosts simultaneously in the proposed model is called for short in the abbreviation form JO which means joint optimization algorithm. The single-objective algorithms which are host optimization (HO) and delay optimization (DO), respectively, are also analysed. HO hosts the service in the best way to saving the number of hosts in meeting the network demand, regardless of the delay caused by the service; DO seamlessly hosts the best service to minimize delay, without considering the number of hosting services. □

5. Results and Discussion

In this section, the ITS application scenery in the simulation scenario is the coverage area of fixed RSUs in the downtown area of Longhua District, Shenzhen, China, where there are totally ten RSUs ($N = 10$) in the topology, as shown in Figure 3.

The network topology shown above represents the possible paths among the ten RSUs which can be utilized to configure the allocation of the cloud resources when it comes to matching the demands of the vehicles. The left part of Figure 3 is the scenario map of the simulated area, while the right part describes the detailed topology of the designed SDN RSUs network. In Figure 3, each RSU is decomposed into a OpenFlow switch and a server, the RSU_0 consists of s_0 and h_0, so for the rest RSUs. It is assumed that the paths represented in a list from origin RSU m to destination RSU n are acyclic elementary chains, where the labelled notations are from 0 to 9. For the topology of the RSU cloud, it is easy to work out that there are 500 possible paths from origin RSU m to destination RSU n, $\forall 0 \le m, n < N = 10$ in summary. From what is shown in Figure 3, considering a confined geographical region, vehicles are entering and leaving the region continuously and the RSUs are deployed along the roads. Taking an example of the flow

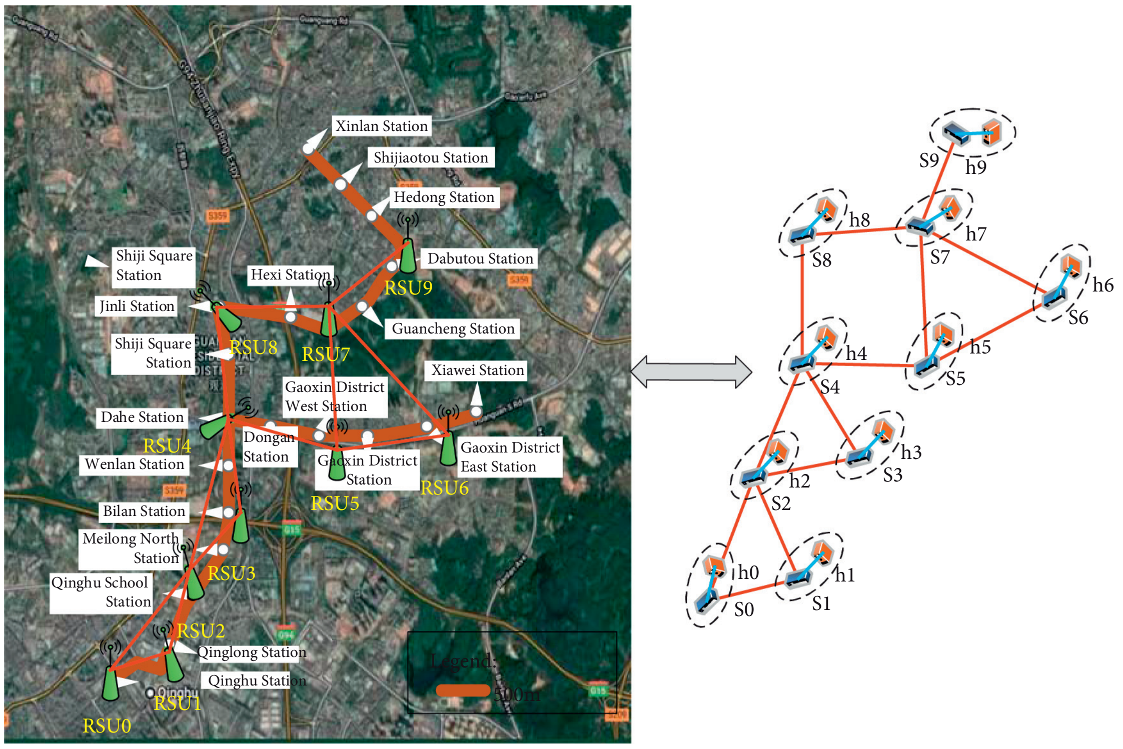


Figure 3: Detailed SDN RSU cloud topology.

$(S_0 - h_0) \longrightarrow (S_9 - h_9)$, which means the traffic flow from RSU_0 to RSU_9, there are totally 12 possible different paths for the flow: $(0, 9, \text{Path}_0)$: [0, 1, 2, 3, 4, 5, 6, 7, 9], $(0, 9, \text{Path}_1)$: [0, 1, 2, 3, 4, 5, 7, 9], $(0, 9, \text{Path}_2)$: [0, 1, 2, 3, 4, 8, 7, 9], $(0, 9, \text{Path}_3)$: [0, 1, 2, 4, 5, 6, 7, 9], $(0, 9, \text{Path}_4)$: [0, 1, 2, 4, 5, 7, 9], $(0, 9, \text{Path}_5)$: [0, 1, 2, 4, 8, 7, 9], $(0, 9, \text{Path}_6)$: [0, 2, 3, 4, 5, 6, 7, 9], $(0, 9, \text{Path}_7)$: [0, 2, 3, 4, 5, 7, 9], $(0, 9, \text{Path}_8)$: [0, 2, 3, 4, 8, 7, 9], $(0, 9, \text{Path}_9)$: [0, 2, 4, 5, 6, 7, 9], $(0, 9, \text{Path}_{10})$: [0, 2, 4, 5, 7, 9], and $(0, 9, \text{Path}_{11})$: [0, 2, 4, 8, 7, 9].

5.1. Parameter Settings. To analyse the experimental results in a representative way, the number of services is set to $S = 1, 2$, which typically represent the single-threading and the multithreading instances, that is, the simple and complex scenario for services, respectively. During period t_1, the average demand D_{t_1} is composed of a multiple of demands $b_{n,k} \cdot \zeta_{n,k}$ at node n for service k. Since that there are many cases in nature that conform to the normal distribution, it can describe a common distribution of samples affected by a large amount of small random disturbances. In this case, the normal distribution is adopted here, similar to the experiments in [19]. For each group of the experiments, the normal distribution is applied with the configuration that the mean value equals to the D (demand), and the standard deviation of the distribution equals to 0.05 * demand. Under this circumstance, one moment of $b_{n,k} \cdot \zeta_{n,k}$ is taken out with $S = 1, 2$, respectively, at 50 and 60 Mbps on behalf of all the average demands ranging from 0 Mbps to 99 Mbps.

In the preliminary experiment, the size of one single packet for the test experiment is 1240 bytes and multiple times are tried with a bunch of same service demands ranging from 0 Mbps to 99 Mbps for each service until there is no feasible solution and $\zeta_{n,k}$ is set to 1 or 0 randomly with the random function of NumPy [26]. In this way, the demands for the services are around D or 0 otherwise, which means the demand differences among the 10 RSUs are great and the SDN RSU cloud resources can therefore be managed.

For $S = 1$, 86 Mbps is the largest flow demand with feasible solution and 97 Mbps is the largest flow demand with feasible solution for $S = 2$, where the entire SDN RSUs cloud is working in saturating state. As a matter of fact, when the flow demands are lower than 30 Mbps, the results of delay and number of hosts change to a small extent, and this is due to the reason that the flow demands are too small. Furthermore, the model is in a multiple dimension when it comes to the circumstance of $S \geq 2$. While $S > 2$ brings the model with nothing but making the network configuration more complex and hard to show the result analysis, therefore $S = 2$ is set when it comes to the circumstance of complex scenario for services. Seven different experimental flow demands $D = \{40\text{ Mbps}, 50\text{ Mbps}, 60\text{ Mbps}, 70\text{ Mbps}, 80\text{ Mbps}, 90\text{ Mbps}, 97\text{ Mbps}\}$ are set for $S = 2$, and $D = \{30\text{ Mbps}, 40\text{ Mbps}, 50\text{ Mbps}, 60\text{ Mbps}, 70\text{ Mbps}, 80\text{ Mbps}, 86\text{ Mbps}\}$ for $S = 1$. All the initial network configuration parameters are shown in Table 3, which depicts all the numerical parameters that have impacts on the experimental results in detail.

As what can be shown in Table 3, the bandwidth capacity of each edge is set to 100 Mbps, and RSU's threshold on number of service hosts is 5. Besides, for one service, its

Table 3: Initial network configuration parameters.

Parameters	Values
C_e	100 Mbps
ϖ_m	5, $\forall 0 \le m < N$.
ϖ_k	5, $\forall 0 \le k < S$.
φ_k	60 ms, $\forall 0 \le k < S$.
ξ	1
$\pi^k_{m,n,j}$	0, $\forall 0 \le k < S$, $0 \le j < \Omega_{m,n}$, $0 \le m, n < N$.
R	1.5

threshold on number of hosts is also set to 5. For the convenience of splitting the network load, the ξ is set to 1, which can split the load into smallest pieces. The initial network configuration assumes that there is no control plane in the network. For the redundancy of the network, R is set to 1.5.

5.2. Experimental Results. In this section, the experiment data are worked out and the results of the cloud delays and hosts numbers for all algorithms are analysed, respectively. To begin with, for the topology of the RSU cloud, the sum of cloud delay refers that all the delays summed for 500 paths from origin RSU m to destination RSU n, $\forall 0 \le m, n < N = 10$ during period t_1.

For $S = 1$, the hosts numbers of joint optimization and delay optimization are all the same with the demands arising from 30 to 86 Mbps. What is more, for $S = 1$, the sum of cloud delay of JO and DO is almost the same as shown in the right part of Figure 4. The above results show that when the service is set to 1, the joint optimization and delay optimization algorithm are the same in essence. On the other side, when compared with the host optimization algorithm, though the number of hosts is less than the other two algorithms, the sum of cloud delay increases explosively.

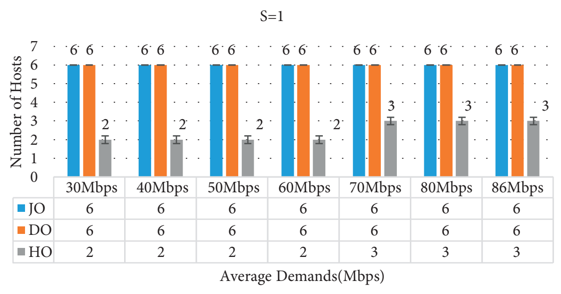
As shown in the left part of Figure 4, when the average demands are below 60 Mbps, the number of hosts can be set as only 2 in HO, greatly less than 6 in JO and HO, which reduces 66.67% number of hosts. What is more, when the average demands are below 80 Mbps, the sums of cloud delay in JO and DO are all much less than HO. When the average demand comes to 86 Mbps which are the maximum bearable demands for each service in the SDN RSU cloud, the sums of cloud delay for all 3 algorithms increase substantially. Though the HO could handle the 86 Mbps situation with only 3 hosts, the sum of delay comes to 8144 ms which is greatly larger than 2961.10 ms in JO and DO. To step further, the HO algorithm represents the minimum number of hosts needed for the SDN RSU cloud.

$S = 2$ represents the complex scenario for services, and the joint optimization algorithm would distinguish from the delay optimization algorithm and host optimization algorithm which is different from the circumstance of $S = 1$. From what can be seen in Figure 5, when the average demands are below 90 Mbps, the JO could achieve near the same sum of cloud delay as DO with the same or even less hosts. Taking 40 Mbps as an example, the sum of cloud delay in JO is 319.40 ms which is only 2.10 ms greater than DO's while the number of hosts in JO is 8 which is 3 less than DO whose number of hosts is 11. For the 40 Mbps circumstance, the delay of JO only increases 0.66% than DO, but the number of hosts in JO decreases 27.27% than JO, which is of great significance for the improvement of the overall cloud efficiency.

From Figure 5, it can be observed that the sum of cloud delays rises at an explosive speed and reaches to over 8000 ms when the network demand arrives at 97 Mbps. The reason why delay grows up drastically when communication demand grows up slightly from 90 Mbps to 97 Mbps accounts for the transfer of the network status. When the network demand is 90 Mbps, the network status is far away from being saturated. However, network status becomes saturated when the network demand comes to 97 Mbps. If the clients and servers are connected with the limited 100 Mbps bandwidth, but there are multiple applications that need to transmit data approximating the bandwidth at the same time, a large amount of data will be waiting in a queue at this time, which makes the delay increase greatly. Furthermore, it can be easily seen that the sum of cloud delays is rising with the demands for the JO and DO algorithms which is in accordance with the normal recognition. As the average demands for each service arise from 40 to 97 Mbps, the number of hosts is rising in HO. However, things get a little bit complex when it comes to the sums of cloud delay in HO. The sum of cloud delay at 50 Mbps in HO is 31.26% larger than 60 Mbps in HO, while the number of hosts of these two is the same. To explain the phenomena, the network configurations Z_{t_1} must be further taken out, which is computed from the HO algorithm in contrast to the initial network configurations Z_{t_0}. The network configurations of the HO algorithm in 50 Mbps and 60 Mbps are shown in Table 4.

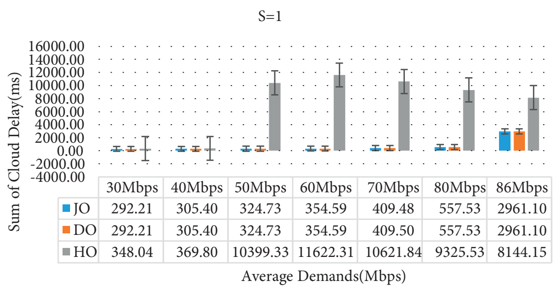
When $S = 2$, in Figure 5, the numbers of hosts for 50 Mbps and 60 Mbps equal to 5, where there are 2 hosts running idly for back up and they do not afford any loads. This is due to the reason that $R = 1.5$ is set to ensure a certain amount of service resource redundancy and $[3 \cdot 1.5] = 5$. Therefore, the two idling hosts in Table 4 are removed for the convenience of analysing the network configurations of HO algorithm. For $S = 2$, as shown in Table 4, when the average demands arrive at 50 Mbps and 60 Mbps in HO algorithm, $h_{2,0} = 1$ and $h_{2,1} = 1$ mean there are two services hosted on RSU_2 at 50 Mbps, while the 3 hosts are distributed on different RSUs at 60 Mbps. Due to the reasons depicted above, there are only 2 RSUs working at 50 Mbps, in contrast to 3 RSUs working at 50 Mbps, leading to the fact that the sum of cloud delay at 50 Mbps is greater than at 60 Mbps with the same number of hosts in HO algorithms. What is more, when it comes to designing the RSU cloud, the above phenomenon reminds us that the services should be deployed globally in a distributed manner rather than in the centralized manner for achieving the minimized cloud delay. The services are recommended distributed rather than concentrated on some specific RSUs.

5.2.1. The Analysis of Network Configurations. Since the sum of cloud delay and number of hosts for JO, DO, and HO



	30Mbps	40Mbps	50Mbps	60Mbps	70Mbps	80Mbps	86Mbps
🟦 JO	6	6	6	6	6	6	6
🟧 DO	6	6	6	6	6	6	6
◼ HO	2	2	2	2	3	3	3

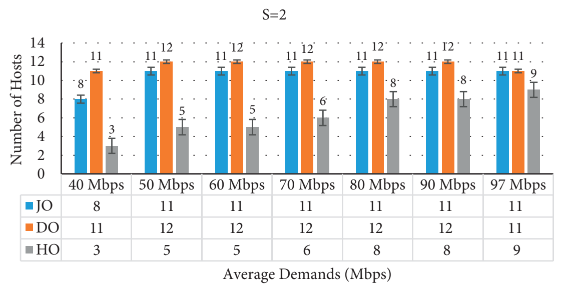
(a)



	30Mbps	40Mbps	50Mbps	60Mbps	70Mbps	80Mbps	86Mbps
🟦 JO	292.21	305.40	324.73	354.59	409.48	557.53	2961.10
🟧 DO	292.21	305.40	324.73	354.59	409.50	557.53	2961.10
◼ HO	348.04	369.80	10399.33	11622.31	10621.84	9325.53	8144.15

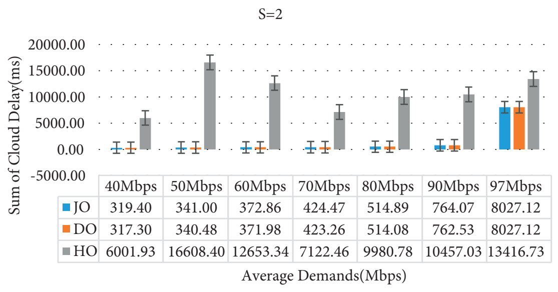
(b)

Figure 4: Global comparison for each algorithm when $S = 1$.



	40 Mbps	50 Mbps	60 Mbps	70 Mbps	80 Mbps	90 Mbps	97 Mbps
🟦 JO	8	11	11	11	11	11	11
🟧 DO	11	12	12	12	12	12	11
◼ HO	3	5	5	6	8	8	9

(a)



	40Mbps	50Mbps	60Mbps	70Mbps	80Mbps	90Mbps	97Mbps
🟦 JO	319.40	341.00	372.86	424.47	514.89	764.07	8027.12
🟧 DO	317.30	340.48	371.98	423.26	514.08	762.53	8027.12
◼ HO	6001.93	16608.40	12653.34	7122.46	9980.78	10457.03	13416.73

(b)

Figure 5: Global comparison for each algorithm when $S = 2$.

Table 4: Network configurations of HO algorithm in 50 Mbps and 60 Mbps when $S = 2$.

50 Mbps	$h_{2,0} = 1$	$h_{2,1} = 1$	$h_{6,0} = 1$	$h_{m,k} = 0\,(h_{m,k} \neq h_{2,0}, h_{2,1}, h_{6,0}, \forall 0 \le m \le 9, 0 \le k \le 1)$
60 Mbps	$h_{1,0} = 1$	$h_{2,1} = 1$	$h_{6,0} = 1$	$h_{m,k} = 0\,(h_{m,k} \neq h_{1,0}, h_{2,1}, h_{6,0}, \forall 0 \le m \le 9, 0 \le k \le 1)$

Table 5: Network configurations of JO, DO, and HO algorithms in 60 Mbps when $S = 1, 2$.

Instances $(S = 1)$	JO		DO		HO		Instances $(S = 2)$	JO		DO		HO	
	50	60	50	60	50	60		50	60	50	60	50	60
$h_{0,0}$	0	0	0	0	0	0	$h_{0,0}$	0	0	0	0	0	0
$h_{1,0}$	0	0	0	0	0	0	$h_{0,1}$	0	0	0	0	0	0
$h_{2,0}$	1	1	1	1	0	0	$h_{1,0}$	1	1	1	1	0	1
$h_{3,0}$	0	0	0	0	0	0	$h_{1,1}$	0	0	0	0	0	0
$h_{4,0}$	1	1	1	1	1	1	$h_{2,0}$	0	0	0	1	1	0
$h_{5,0}$	1	1	1	1	0	0	$h_{2,1}$	1	1	1	1	1	1
$h_{6,0}$	1	1	1	1	0	0	$h_{3,0}$	1	1	1	1	0	0
$h_{7,0}$	0	0	0	0	0	0	$h_{3,1}$	0	0	0	0	0	0
$h_{8,0}$	0	0	0	0	0	0	$h_{4,0}$	0	0	0	0	0	0
$h_{9,0}$	0	0	0	0	0	0	$h_{4,1}$	0	0	0	0	0	0
							$h_{5,0}$	0	0	0	0	0	0
							$h_{5,1}$	0	0	0	0	0	0
							$h_{6,0}$	1	1	1	1	1	1
							$h_{6,0}$	0	0	0	0	0	0
							$h_{7,0}$	0	0	0	0	0	0
							$h_{7,1}$	1	1	1	1	0	0
							$h_{8,0}$	1	1	1	0	0	0
							$h_{8,1}$	0	0	1	1	0	0
							$h_{9,0}$	1	1	1	1	0	0
							$h_{9,1}$	0	0	0	0	0	0

algorithms in 60 Mbps when $S = 1, 2$ are already displayed in Figures 4 and 5, here the idling hosts are removed for the convenience of analysing the network configurations. To further demonstrate the reasons why there are different sums of cloud delay for the 3 different optimization algorithms, the working hosts' configurations are shown in detail at Table 5.

From what can be seen in the left part of Table 5, the network configurations of JO and DO algorithms in 50 and 60 Mbps when $S = 1$ are exactly the same, which further verifies that these two algorithms are the same in essence when $S = 1$. Meanwhile, when $S = 2$ at 50 Mbps, JO algorithm needs 1 less host than DO algorithm where their other host network configurations are the same, except for the $h_{8,1}$. That is, while the $h_{8,1}$ in JO is 0, the $h_{8,1}$ is set to 1 in DO. Under this circumstance, the sum of cloud delay in JO is 341.00 ms while 340.48 ms in DO, showing that JO could achieve almost the same sum of cloud delay as DO even with 1 less host. For the HO results compared with the JO and DO, though the number of hosts is reduced to a great extent, the sum of delay is much larger than JO and DO and unsuitable for use in the application. Still, it gives us the emergency resources management solution in the SDN RSU cloud when the network congestion happened.

6. Conclusions and Future Works

To reduce environmental pollution, achieving a full range of safe, efficient, and green transportation environment, VANETs have been regarded as a key enabling technology for improving the road safety and building intelligent transportation systems. VANETs can provide safety and nonsafety-related applications and services to improve the safety and shared mobility of vehicle users. It is a multidisciplinary technical field involving road traffic, wireless communication, self-organizing systems, etc. The cloud resource management system realizes the interoperability of resource data in the VANETs. It can integrate similar or related resources through resource integration services to reduce the transmission of redundant data. On this basis, through resource data distribution planning services, according to the priority of the resource, under the condition of limited bandwidth, the priority transmission of key data is realized.

In this paper, the deep programmability of SDN is used to dynamically reconfigure the network for hosting, routing the services, and effectively meeting the basic requirements of the VANETs. Multipath is also enabled to forward data streams to balance the loads on the network. To achieve the minimum cloud delay with the least number of hosts, it can be summarized as a mixed-integer linear programming problem (MILP). Based on the research of the architecture technology in cloud resource management system, a cloud resource management system for vehicle collaborative perception scenarios is realized, and the feasibility of the system is verified. When the service is set to 1, which typically represents the single-threading instance scenario, the common optimization and delay optimization algorithms are essentially the same. Besides, the results show that, given the requirements, when the services are larger or equal to 2, the proposed algorithm is superior to single-objective optimization algorithms, namely, the delay optimization (DO) algorithm and the host optimization (HO) algorithms. In addition, the results also indicate that services should be distributed on a global scale, rather than provide services centrally.

For future work, the proposed algorithm can be used to the shared mobility services in connected and automated environment to improve the safety. The inclement road conditions can have impact on the traffic safety, level of service, traffic mobility, and fuel efficiency of the vehicle flows [27], and its communication mode can be further studied and combined with our proposed model in the future research. Besides, the cloud resource management system could be enhanced by comparing it with the centralized resource management system under the scenario of unmanned vehicle collaborative perception. Furthermore, in terms of RSU access, more complex scenarios can be considered in follow-up research, such as overlapping coverage of multiple RSUs, variable speed of vehicles can also be considered. Such in-depth research will be more practical. The follow-up work can further study this special circumstance and combine with other advanced algorithms to propose more effective data distribution algorithms. There are also lots of security and privacy issues and research challenges in RSU clouds.

Data Availability

The data used to support the findings of this research are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest for the publication of the paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (52172329) and the 13th Five-Year National Key Research and Development Plan (2018YFB1201403).

References

[1] E. K. Adanu and S. Jones, “Effects of human-centered factors on crash injury severities,” *Journal of Advanced Transportation*, vol. 2017, no. 2017, 11 pages, Article ID 1208170, 2017.

[2] Z. Pu, Z. Li, Y. Jiang, and Y. Wang, “Full Bayesian before-after analysis of safety effects of variable speed limit system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 964–976, 2021.

[3] D. Sun, Y. Zheng, and R. Duan, “Energy consumption simulation and economic benefit analysis for urban electric commercial-vehicles,” *Transportation Research Part D: Transport and Environment*, vol. 101, Article ID 103083, 2021.

[4] W. Li, Z. Pu, Y. Li, and M. Tu, “How does ridesplitting reduce emissions from ridesourcing? a spatiotemporal analysis in

Chengdu, China," *Transportation Research Part D: Transport and Environment*, vol. 95, Article ID 102885, 2021.

[5] M. Tu, Y. Li, W. Li, M. Tu, O. Orfila, and D. Gruyer, "Improving ridesplitting services using optimization procedures on a shareability network: a case study of Chengdu," *Technological Forecasting and Social Change*, vol. 149, Article ID 119733, 2019.

[6] D. Ou, Y. Yang, L. Xue, and D. Dong, "Optimal connectivity-based deployment of roadside units for vehicular networks in urban areas," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2559, no. 1, pp. 46–56, 2016.

[7] H. Hartenstein and K. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 6, pp. 164–171, 2008.

[8] C. Campolo, A. Molinaro, A. Vinel, N. Lyamin, and M. Jonsson, "Service discovery and access in vehicle-to-roadside multi-channel VANETs," in *Proceedings of the IEEE International Conference on Communication Workshop*, pp. 2477–2482, London, UK, June 2015.

[9] K. Mershad and H. Artail, "Finding a STAR in a vehicular cloud," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 55–68, 2013.

[10] F. S. Behbehani, T. Elgorashi, and J. M. H. Elmirghani, "Power minimization in vehicular cloud architecture," pp. 1–32, 2021, https://arxiv.org/abs/2102.09011.

[11] A. Chinnasamy, B. Sivakumar, P. Selvakumari, and A. Suresh, "Minimum connected dominating set based RSU allocation for smartCloud vehicles in VANET," *Cluster Computing*, vol. 22, no. s5, pp. 12795–12804, 2019.

[12] Q. Wang, P. Fan, and K. B. Letaief, "On the joint V2I and V2V scheduling for cooperative VANETs with network coding," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 62–73, 2012.

[13] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[14] C. Tang, C. Zhu, X. Wei, H. Wu, Q. Li, and J. J. P. C. Rodrigues, "Intelligent resource allocation for utility optimization in RSU-empowered vehicular network," *IEEE Access*, vol. 8, pp. 94453–94462, 2020.

[15] Q. Zheng, K. Zheng, P. Chatzimisios, and F. Liu, "Joint optimization of link scheduling and resource allocation in cooperative vehicular networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 179, 2015.

[16] E. Lee, E.-K. Lee, M. Gerla, and S. Oh, "Vehicular cloud networking: architecture and design principles," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 148–155, 2014.

[17] S. Olariu, M. Eltoweissy, and M. Younis, "Towards autonomous vehicular clouds," *ICST Transactions on Mobile Communications and Applications*, vol. 11, no. 7–9, p. e2, 2011.

[18] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM-Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[19] 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.

[20] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing IEEE 802.11p/WAVE to provide infotainment applications in VANETs," *Ad Hoc Networks*, vol. 10, no. 2, pp. 253–269, 2012.

[21] M. A. Salahuddin, A. Al-Fuqaha, M. Guizani, and S. Cherkaoui, "RSU cloud and its resource management in support of enhanced vehicular applications," in *Proceedings of the 2014 IEEE Globecom Work GC Workshops*, pp. 127–132, San Diego, CA, USA, December 2014.

[22] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 2010 IEEE INFOCOM0*, pp. 1–9, San Diego, CA, USA, March 2010.

[23] G. L. Curry and R. M. Feldman, *Manufacturing Systems Modeling and Analysis*, Springer, Berlin Heidelberg, Germany, 2009.

[24] A. P. Wierzbicki, "A methodological guide to multi-objective optimization," in *Proceedings of the Optimization Techniques, Part 1*, Vol. 22 of Lecture Notes in Control and Information Sciences, K. Iracki, K. Malanowski, and S. S. Walukiewicz, eds., Springer, Berlin, Germany, 1980.

[25] D. Li, Y. Yu, W. He, K. Zheng, and B. He, "Willow: saving data center network energy for network-limited flows," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2610–2620, 2015.

[26] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[27] Z. Pu, C. Liu, X. Shi, Z. Cui, and Y. Wang, "Road surface friction prediction using long short-term memory neural network based on historical data," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, pp. 1–12, 2020.