Received February 20, 2021, accepted February 27, 2021, date of publication March 2, 2021, date of current version March 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063246

Virtual Edge: Exploring Computation Offloading in Collaborative Vehicular Edge Computing

NARISU CHA[1], CELIMUGE WU[1], (Senior Member, IEEE), TSUTOMU YOSHINAGA[1], (Member, IEEE), YUSHENG JI[2], (Senior Member, IEEE), AND KOK-LIM ALVIN YAU[3], (Senior Member, IEEE)

[1]Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu 182-8585, Japan
[2]Information Systems Architecture Research Division, National Institute of Informatics (NII), Tokyo 101-8430, Japan
[3]School of Science and Technology, Sunway University, Subang Jaya 47500, Malaysia

Corresponding author: Celimuge Wu (celimuge@uec.ac.jp)

This work was supported in part by the ROIS National Institute of Informatics Open Collaborative Research under Grant 2020-20S0502, and in part by the JSPS KAKENHI under Grant 18KK0279, Grant 19H04093, and Grant 20H00592.

ABSTRACT Vehicular edge computing (VEC) has been a new paradigm to support computation-intensive and latency-sensitive services. However, the scarcity of computational resources is still a challenge. Making efficient use of sporadic idle computational resources on smart vehicles in the vicinity to extend the resource capability of each vehicle is an important research issue. In this paper, we propose Virtual Edge, which is an efficient scheme to utilize free computational resources of multiple vehicles as a virtual server to facilitate collaborative vehicular edge computing. We design a virtual edge formation algorithm that considers both the stability of virtual edge and the computational resources available at the vehicles constituting the virtual edge. The prediction of the link duration between vehicles reduces the number of computation offloading failures caused by unexpected link disconnections. Extensive simulations with realistic vehicle movements are conducted to show the advantage of the proposed scheme over existing baselines in terms of the completion ratio of computation offloading tasks and average task execution time.

INDEX TERMS Collaborative vehicular edge computing, virtual edge, computation offloading.

I. INTRODUCTION

The emergence of vehicular edge computing (VEC) has been a new paradigm to support the ever-growing computation-intensive and latency-sensitive services in intelligent transportation systems (ITS) [1]. However, there are still some challenges in conducting efficient edge computing tasks due to the high mobility of vehicles and the limited bandwidth in vehicular networks. Smart vehicles are also referred to as "computers with the wheels", which are equipped with communication devices to connect with each other, and have powerful processing unit to execute computation tasks. It is forecasted by IHS Markit that the number of vehicles on the roads and in-vehicle equipment could reach nearly two billion by 2025, and each car could produce up to 30 terabytes of data every day [2]. On the other hand, the workloads in the vehicles are not evenly distributed. First, slow-moving vehicles have less computation tasks [3] in a relatively stable road traffic context. In contrast, fast-moving vehicles have to process a much larger amount of data in a shorter time to guarantee a safe driving. Second, for some specific scenarios, such as vehicle platoons, some vehicles conduct computation tasks (platoon leader) and inform the results to other vehicles (platoon members) through vehicle-to-vehicle communications. There could be also some differences in computational capabilities for different vehicles. Therefore, the problem of "how to efficiently utilize the idle vehicle computational resources" needs further investigations.

Most existing studies discuss about the offloading of computation tasks from vehicles to roadside units (RSU), cloud servers, or a single vehicle. In this paper, we discuss how to offload tasks to multiple vehicles. In order to conduct efficient task offloading to multiple vehicles, the communication capability between the vehicles should be considered. This is because a task requester vehicle must allocate its tasks to neighboring vehicles that aggregate their computed results in a collaborative manner. Existing studies split each task into multiple jobs and basically use a reactive approach to allocate

The associate editor coordinating the review of this manuscript and approving it for publication was Ivan Wang-Hei Ho.

jobs to multiple vehicles after the task offloading request is made. However, the reactive approach has a fatal problem that the task requester must first perceive the environment and then conduct task offloading, and so a certain amount of time must be incurred before decisions are made. Moreover, since each task requester makes decision independently, the global optimum is difficult to achieve.

In this paper, we propose a proactive task offloading framework, namely, virtual edge, to solve the task offloading problem in vehicular environments. Each virtual edge node consists of multiple vehicles with low relative moving speed and good wireless connectivity between them. It enables the vehicles to share free computational resources and provide task execution functionality to other vehicles. By efficiently utilizing the computational resources of multiple vehicles, the virtual edge is able to provide a much powerful computational capability than a single vehicle could provide. When a vehicle needs to offload computation tasks to other vehicles, it constructs a new virtual edge and offloads the tasks to it, or offloads the tasks to a virtual edge in the vicinity. By using the concept of the virtual edge, an efficient virtual computing node can be constructed, resulting in a much efficient task offloading as compared with the conventional reactive task offloading approaches.

In order to provide a satisfactory service to task requesters, a virtual edge must be a relatively stable unit that does not vary too much in terms of the number and capabilities of the members. However, the formation of virtual edge is a complex problem. Due to the mobility of vehicles, there could be frequent changes in the members of a virtual edge, which incurs a negative impact on the virtual edge performance. Therefore, the number of members should be limited, and the members should have stable connections with each other. In other words, the vehicle velocity, the wireless connectivity between vehicles, and computational capability of vehicles should be considered in the virtual edge formation. In this paper, we propose virtual edge to facilitate a collaborative vehicular edge computing environment. Our main contributions in the paper are as the following.

- We propose a new scheme, namely, virtual edge, which is formed by multiple vehicles with available computational resources in the vicinity, for facilitating collaborative vehicular edge computing. The virtual edge scheme can use multiple vehicles in vicinity to enhance task processing in vehicular networks.
- We propose an efficient algorithm to form a virtual edge. In order to predict the time duration of a virtual edge, we introduce a new metric, namely, the companion time among the members of the virtual edge. The companion time and computational resources are considered in the formation of the virtual edge.
- Extensive computer simulations with realistic vehicle mobility are conducted, and the effectiveness of the proposed approach is shown by comparing it with three existing baselines. By considering the connection duration time and computational capability at each vehicle, the proposed scheme is able to provide a much higher performance as compared with existing baselines.

This paper is an extension of our previous conference paper [4]. While [4] only discusses the problem of how to establish a virtual edge, this paper includes all the details of task offloading using virtual edge, and shows new simulation results regarding the task offloading performance. The remainder of this paper is as follows. Related studies are summarized in section II, including computation offloading and the collaboration among vehicles in vehicular edge computing. Then, we briefly describe the architecture of virtual edge in section III. In section IV, we present system model and problem formulation. In section V, the proposed scheme is described in detail, and experimental results are demonstrated in section VI. Finally, conclusion and future work are drawn in section VII.

II. RELATED WORK

A. COMPUTATION OFFLOADING FOR VEHICULAR EDGE COMPUTING

Improving task execution performance in computation offloading is one of the key challenges in multi-access edge computing (MEC). To face the challenge, researchers conduct many kinds of studies, and the results are mainly focused on the energy-efficient allocation of computing resources [5], binary computation offloading [6], and partial computation offloading [7], [8]. In [9], [10], the authors systematically summarize the recent efforts on investigations related to computation offloading in MEC. Most of the existing studies only consider the offloading computation tasks from vehicles to edge servers, such as roadside units (RSU) or cloud servers, and the problem of how to offload computation tasks to multiple moving vehicles has not been clarified.

Smart vehicles on the road can be viewed as edge nodes with computational resources to execute driving tasks, such as 3D traffic scene generation, lane identification, positioning, image processing, and traffic sensing. Multiple smart vehicles can collaborate with each other to accomplish collaborative tasks in some scenarios where existing infrastructure cannot support the tasks with adequate resources. These tasks feature intensive computations that can be offloaded to some vehicles in the vicinity. In [10], Mao *et al.* introduce a method of using joint radio and computational resource management to merge the two disciplines of wireless communications and mobile computing seamlessly. Some studies [11]–[13] conceive the idea of vehicles as an infrastructure (VaaI) where moving and parked vehicles with idle resources are viewed as infrastructures for communications and computing, and analyze the feasibility of VaaI. Hou *et al.* [11] concludes that adjacent moving vehicles have good connectivity with one another and they can form a powerful computation cluster. They describe four kinds of application scenarios and a three-layer vehicular fog computing (VFC) paradigm, then verify them using real-world vehicular mobility traces. The results indicate that

VFC can increase the availability and capability of resources. Ning *et al.* [14] propose a decentralized three-layer VFC architecture, where moving and parked cars are viewed as fog nodes for processing local traffic data to balance the network load. Vehicular MEC with collaborative task offloading can guarantee low latency of the applications [15]. The optimal policy for computation offloading to multiple base stations in an ultra-dense sliced RAN is modeled by Markov decision process, and a new algorithm to offload computation tasks based on double deep Q-network is proposed in [16]. To tackle the spatial intelligence of vehicular Internet of Things (IoT), the technology of using Q-learning to combine decentralized moving edge nodes with multi-tier multi-access edge clusters is discussed in [17]. Sun *et al.* in [18] develop an algorithm to exploit the abundant resource on vehicles. There are also some discussions on joint resource allocation and computation offloading [19]–[21]. However, how to utilize the computing capabilities of moving vehicles efficiently in dynamic vehicular networks has not been adequately discussed in existing studies.

B. COLLABORATIVE VEHICULAR EDGE COMPUTING

Collaborative vehicular edge computing (CVEC) has been studied in many articles covering collaborative coalition formation [22], collaborative computation offloading between vehicles [23], volunteer assisted collaborative offloading [24], and collaborative route selection [25]. Wang *et al.* [26] consider using edge servers with both horizontal and vertical collaborative offloading to support vehicular services in CVEC. While these studies focus on how to optimize computation offloading in some specific scenarios, this paper discusses how to form a stable virtual edge node, which can be used for meeting different types of computation offloading requirements.

Compared with cloud computing, devices available in CVEC, such as vehicle nodes, are resource-limited. These devices have the following characteristics: sporadic resource availability, high mobility, and high heterogeneity. Therefore, one of the core problems of CVEC is how to orchestrate the resources and computation efficiently in the edge nodes. Due to the vehicle movement, vehicular services on the edge must be migrated promptly and regularly to guarantee a constrained latency. The study [9] considers virtual machines (VMs) as the virtualization platform. However, the overhead incurred by the VMs has not been addressed. Due to the high mobility of vehicles, the authors foresee a demand on a kind of lightweight container (a virtualization technology to achieve collaborative computing) with the following requirements: to rapidly reconfigure the switching context, to seamlessly orchestrate the allocation of resources, and to migrate task execution state to a new location. Recently, container-based task execution has drawn great attention of many researchers because of its lightweight feature as compared with VMs [27]. Alam *et al.* [28] combine Docker, which is a platform using container based virtualization technology, with edge computing micro-service to execute IoT applications in different layers. Tang *et al.* in [29] develop a container, which is deployed on the edge server with abstraction and management functions. Compared with the studies mentioned above, Huang *et al.* in [30] consider a scenario where each vehicle is installed with a container, and the parking lots are equipped with a backbone of VEC servers for task offloading in order to acquire a higher overall utility level. What we can conclude from the previous studies is that there is a trend that virtualization technologies are spreading from the core networks to the infrastructure on the edge.

III. VIRTUAL EDGE ARCHITECTURE

The architecture of the classical VEC consists of three layers: the cloud layer, the edge layer, and the user layer. The vehicles, which are in the user layer, can be viewed as mobile nodes with available computational resources in the architecture.

A. CLOUD LAYER

This layer consists of the cloud server to store long-term data. For example, traffic management and trusted third authority data are stored in this layer to monitor global traffic and optimize the entire system.

B. EDGE LAYER

This layer consists of network infrastructures, such as RSU, smart traffic light, and VEC server at the base station (BS). The layer processes data collected from the user layer, and then send it to the cloud server.

C. USER LAYER

All of the user devices with constrained resources belong to this layer, including vehicle, cell phone, and so on. This layer generates raw data.

As shown in Fig. 1, each virtual edge, which lies in the user layer of the conventional VEC architecture, consists of multiple vehicle nodes, including the master vehicles and slave vehicles, as follows.

D. VEHICLE NODE

A vehicle with computational and caching capabilities can serve as an edge node to accomplish the computation tasks. Every vehicle is installed with a container for offloading external tasks. Vehicle nodes are equipped with IEEE 802.11p enabled communication module for exchanging information with other vehicle nodes.

E. MASTER NODE

The master node is the leader of a virtual edge. The master node assigns computation tasks to other members of the virtual edge, namely, slave nodes, as explained below. When a vehicle wants to offload computation tasks to other vehicles, and there is no virtual edge available in the vicinity, the vehicle can work as a master node to establish a new virtual edge. The established virtual edge can be used by other vehicles.

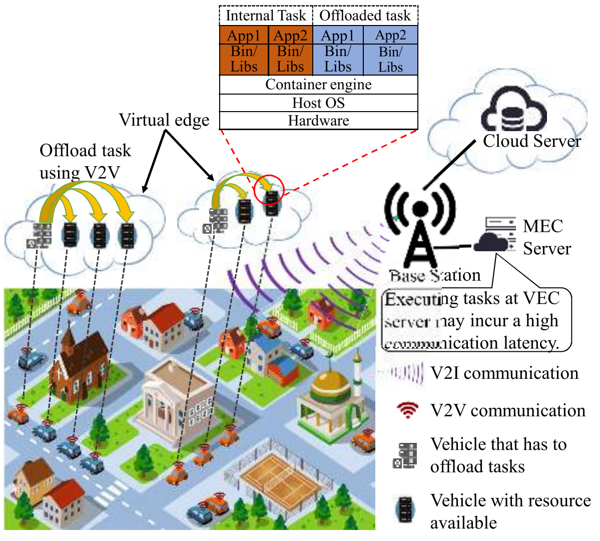

FIGURE 1. Virtual edge and VEC.

F. SLAVE NODE

The slave node is a member of a virtual edge that can provide free computational resources to the master node. A virtual edge may contain multiple slave nodes.

Influenced by the deployment cost and communication range of the underlying infrastructure, the link duration [31] between a vehicle and a MEC server is limited. In this case, using available resources on the vehicles for offloading has many potential advantages. It is generally acknowledged that smart vehicles have more sensing and computational resources than manual driving vehicles. Vehicle nodes with insufficient computational resources, including those who do not obtain computing resources adequately from base stations, offload their computation tasks to other vehicle nodes. The vehicle nodes with tasks may establish an on-demand virtual edge among vehicle nodes with available computational resources in the vicinity. The link duration between the vehicle nodes is predicted and used in the creation of the virtual edge.

A virtual edge is established by vehicle nodes to explore lower-cost resources when the computational resources provided by a VEC server do not satisfy the vehicle node's requirements. This architecture is suitable for dense vehicle traffic, such as urban areas, and it can utilize the computational resources of parked vehicles and/or other vehicle nodes running in the same direction. Without modifying the classical framework of CVEC, virtual edge can be used as a complement to expand the function of utilizing resources in the vicinity. Since the virtual edge is fully compliant with the conventional VEC architecture, a task can be sent to the edge server or cloud server when it cannot be completed at virtual edge.

IV. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

Short-range communication technologies are widely used in vehicle-to-vehicle (V2V) information exchange. Until now, two types of V2V communication technologies are well discussed, namely, IEEE 802.11p and cellular-V2V. While the performance difference of the two types of technologies depends on the distribution of vehicles and traffic flow [32], this paper discusses the use of IEEE 802.11p in V2V communications. According to IEEE 1609.4 [33], each vehicle needs to broadcast a basic safety message (BSM) in a certain time interval. BSM includes vehicle information, such as position, velocity, direction, acceleration, and so on. The standard payload of BSM is 39 bytes. BSM can be further extended to contain the information related to available computational resources like CPU frequency, caching size, etc. Each vehicle maintains a BSM table to keep track of the information of vehicles in the vicinity, and then computes the link duration between two vehicles.

There are $|V(t)|$ vehicles moving on the road to their destinations where one of the vehicles is the master vehicle. Each vehicle is willing to offer its idle computational resources to the master node. Every vehicle follows an intelligent driver model with traffic lights [34] to generate realistic mobility of vehicles in the urban area. A vehicle may offload its computation to the other vehicles when its local resource cannot satisfy the requirement. Master node plays the role of the service requester and the others are service providers. The set of vehicles with available computational resource at time t is shown by $V(t)$, where $V(t) = \{v_1, v_2, \ldots, v_{|V(t)|}\}$, and $|V(t)|$ represents the total number of vehicles at time t. $V_v(t)$ is the set of virtual edge members led by node v at time t and $V_v(t) \subseteq V(t)$, where v represents a node with computation tasks required to be offloaded. Computation tasks can be divided into multiple blocks, which are denoted as $D = \{d_1, d_2, \ldots, d_{|V_v(t)|}\}$, where $|V_v(t)|$ represents the number of virtual edge members. To establish a virtual edge, the master node selects some vehicles based on the information of BSM table (all the information of vehicles in the vicinity is recorded in the table). Then, computation sub-tasks are offloaded to the slave node based on their computational capabilities.

The establishment process of a virtual edge utilizes the information of the BSM table and link duration. As shown in Fig. 2, vehicles within the single-hop communication range show different levels of capabilities. It is assumed that vehicle v_2 has some tasks and needs to offload these tasks to the other vehicles. v_1, v_3, v_4, v_5 are located within the communication range of v_2. Due to the movement of vehicles, the duration of connection between v_2 and v_1 becomes short because two vehicles are driving in the opposite direction. The link between v_2 and v_5 is unstable too because v_5 is located at the border of the transmission range of v_2. Therefore, v_1 and v_5 should not be included in the virtual edge led by v_2.

B. COMMUNICATION MODEL

The computation tasks of the master node are allocated to the slave nodes by using V2V communications. We use the same transmission model in the study [24]. Transmission data

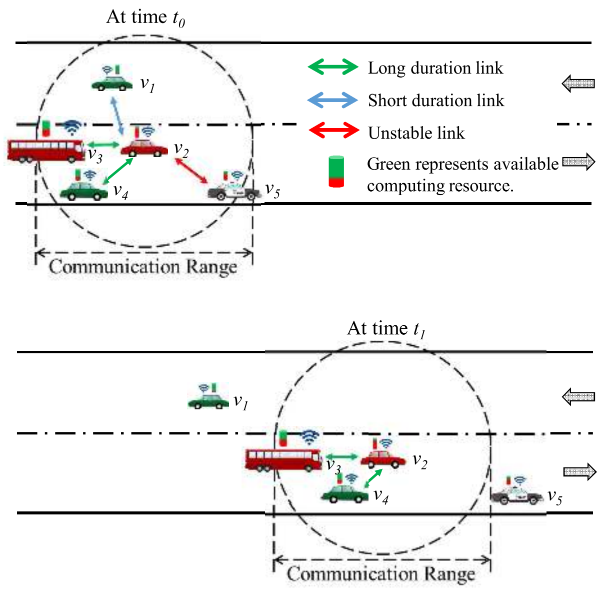

FIGURE 2. **An example of collaborative vehicular edge computing; the link duration and the available computational resources are considered in the establishment of a virtual edge.**

rate $p_{v,u}$ from a master node v to a slave node u is expressed as

$$p_{v,u} = b_0 \log_2(1 + \frac{r_{v,u} h_{v,u}}{\sigma_{v,u}^2}) \tag{1}$$

where $r_{v,u}$ denotes the transmission power from node v at node u, and $h_{v,u}$ is the channel gain. b_0 is the allocated bandwidth when transmitting computation tasks from node v to node u, and $\sigma_{v,u}^2$ represents the additional white Gaussian noise power.

The time for sending computation tasks from the master node v to the slave node u is expressed as

$$t_{tra,u} = \frac{d_u}{p_{v,u}}, \quad d_u \in D. \tag{2}$$

The total transmission time for sending computation tasks from the master node v to all its slave nodes $V_v(t)$ can be calculated as

$$t_{tra,v} = \sum_{\forall u \in V_v(t)} t_{tra,u}. \tag{3}$$

The efficiency of the offloading process is not only affected by the stability of the link between vehicles but also the hop count during data transmissions. We use an experimental result as shown in Fig. 3 to estimate the data transmission time for different numbers of hop counts.

C. COMPUTATION MODEL

In this paper, we consider the task, such as image recognition and video retrieval. This kind of the task can be divided into many sub-blocks. Each sub-block would be sent to a remote server (edge server) for execution. The results will be returned to the original node immediately after the computation is completed. Compared to the input data size, the output data

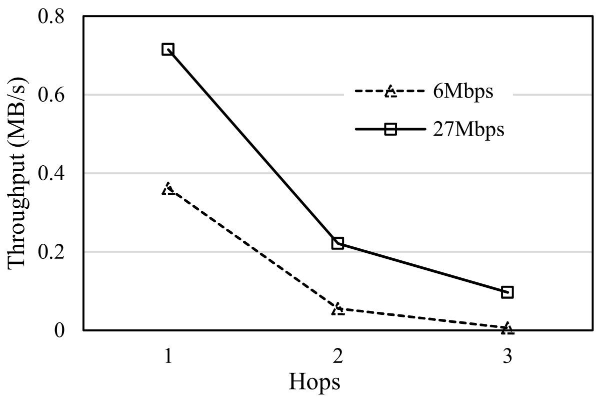

FIGURE 3. **The throughput achieved by using TCP (RENO) for data transmission over V2V communication without interference. The speed of the vehicle is 20km/h, and the data rates are 6 Mbps and 27 Mbps respectively.**

size of the task is very small and the returning (sending back) time of results ($t_{ret,v}$) could be ignored. Therefore, the time for returning results equals to zero in the paper.

The processing time of computation t_u can be expressed as

$$t_u = \frac{\gamma d_u}{\alpha_u f_u}, \quad u \in V_v(t) \tag{4}$$

where the computational resource (i.e., CPU clock frequency) of the vehicle u is expressed as f_u, α_u is the idle resource ratio of the vehicle u, d_u denotes the data size of the block for vehicle u, and γ is service coefficient describing the relationship between data size and the required CPU cycle count for the computation. The members of virtual edge can process the computation simultaneously, and therefore the processing time of the computation is expressed as

$$t_{com,v} = \{\max t_u | \forall u \in V_v(t)\} \tag{5}$$

D. PROBLEM FORMULATION

Our objective is to minimize the execution time of computation tasks on the virtual edge. The execution time consists of three parts: the transmission time $t_{tra,v}$ for sending blocks of computation tasks from the master node u to the slave nodes, the time for processing computation on the slave nodes $t_{com,v}$, and the time for returning (sending back) the computation results from the slave nodes to the master node $t_{ret,v}$. The objective function can be expressed as

$$\begin{aligned} &\min_{V_v(t)} \quad (t_{tra,v} + t_{com,v} + t_{ret,v}) \\ &\text{s.t.} \quad t_{tra,v} + t_{com,v} + t_{ret,v} \leq t_{dur,v} \\ &\qquad V_v(t) \subseteq V(t) \end{aligned} \tag{6}$$

where $t_{dur,v}$ represents the duration of the virtual edge. $V_v(t)$ represents the set of virtual edge members led by the master node v. The constraint denotes that the computation tasks must be processed completely before the virtual edge is broken.

To obtain the minimum processing time of computation, the master node must consider all possible combinations of vehicles in the formation of virtual edge, which are varying with time. Therefore, the above problem is considered a Knapsack problem which has proven to be NP-hard in [35]. To solve the problem, we propose a heuristic approach that jointly considers the duration of virtual edge and computational capability of vehicles. The details of the scheme will be explained in section V.

V. FORMATION OF VIRTUAL EDGE

In order to predict the time duration of a virtual edge, we first introduce a new metric, namely, the companion time, to show the relative mobility level between two vehicles. Then, we explain the implementation process of the proposed scheme. The implementation of virtual edge is illustrated in Fig. 4. First, the master node selects the candidates of slave nodes, and then sends requests to the candidates. After receiving acknowledgments (ACKs) from the candidates, the master node confirms the slave node members. Then, the computation tasks can be offloaded from the master node to the slave nodes.

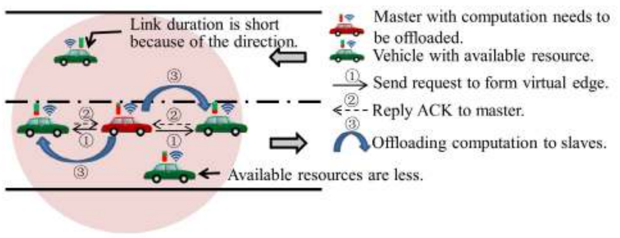
Link duration is short because of the direction.
Master with computation needs to be offloaded.
Vehicle with available resource.
① Send request to form virtual edge.
② Reply ACK to master.
③ Offloading computation to slaves.
Available resources are less.
①
②
③

FIGURE 4. The implementation of virtual edge.

A. DURATION TIME OF VIRTUAL EDGE

The selection process of slave nodes considers the time duration of the virtual edge. Due to the fast mobility of vehicles, it is challenging to choose appropriate vehicle nodes to establish a stable virtual edge. The prediction of the time duration of the virtual edge is important to ensure that the computation results can be returned before the link between vehicles is broken.

Based on the interaction between vehicles, vehicle mobility models in the literature can be categorized into three types: macroscopic, mesoscopic, and microscopic. Macroscopic and mesoscopic models focus on the effect of the traffic parameters, such as the vehicle density and the traffic arrival rate, to the overall performance. Microscopic model describes the mobility of vehicles based on their interactions with neighboring vehicles and the driving behavior of individual vehicles. Here, we use a microscopic approach. In the following, we first introduce three definitions, then define the companion time between two vehicles, and derive the duration of a virtual edge.

Definition 1: Given the assumption that $V(t)$ is the set of vehicle nodes, for node $v, u \in V(t)$, we say that v and u can communicate with each other directly if $dist(v, u) < r$ is satisfied, where $dist(v, u)$ is the distance between node v and u, and r is the communication range of the vehicles. It is denoted by $< v, u >$.

Definition 2: Given the assumption that U is the subset of $V(t)$, if $\forall v, u \in U, \exists v_1, v_2, \ldots, v_n \in U$, satisfy $< v, v_1 >, < v_1, v_2 >, \ldots, < v_{n-1}, v_n >, < v_n, u >$, then it is called that U is interconnected.

Definition 3: Given the assumption that U is the subset of $V(t)$, in the period T, if $\forall t \in [0, T], \forall v, u \in U$, nodes v and u are connected directly, it is called that the duration of U is at least T.

Estimating the duration of subset U encounters some difficulties because of vehicle mobility. Therefore, in the following, we exploit a method to derive the duration of a virtual edge. Here, we introduce the companion time, which is expressed by the upper limit of the link duration between two nodes. The companion time is derived based on a general discretized longitudinal kinematic motion equation of vehicles [34]. The equation of acceleration in this paper is based on the intelligent driver model (IDM) [34]. Each vehicle is only responsible for calculating the companion time of direct connection with its neighborhoods. The vector $\vec{P_v} = (p_{x,v}, p_{y,v}, p_{z,v})$, $\vec{v_v} = (v_{x,v}, v_{y,v}, v_{z,v})$, and $\vec{a_v} = (a_{x,v}, a_{y,v}, a_{z,v})$ denote the position, velocity, and acceleration of vehicle v, respectively. Three elements of each vector represent the values in the x, y, and z directions, respectively. The relative distance of two vehicles is

$$\vec{P} = \vec{P_u} - \vec{P_v} = (p_{x,u} - p_{x,v}, p_{y,u} - p_{y,v}, p_{z,u} - p_{z,v}). \quad (7)$$

Similarly, the relative velocity and acceleration of two vehicles are

$$\vec{a} = \vec{a_u} - \vec{a_v} = (a_{x,u} - a_{x,v}, a_{y,u} - a_{y,v}, a_{z,u} - a_{z,v}) \quad (8)$$

and

$$\vec{v} = \vec{v_u} - \vec{v_v} = (v_{x,u} - v_{x,v}, v_{y,u} - v_{y,v}, v_{z,u} - v_{z,v}). \quad (9)$$

The trajectory of node u relative to node v can be expressed as

$$\vec{S} = \frac{1}{2}\vec{a}\, t^2 + \vec{v}\, t + \vec{P}. \quad (10)$$

Then, the length of relative trajectory can be expressed as

$$\vec{S} \cdot \vec{S} = r^2, \quad (11)$$

where r is the radius of the vehicle's communication range, and $(\cdot)$ denotes the inner product. The formula above is expanded as

$$\begin{aligned}&(\frac{1}{2}(a_{x,u} - a_{x,v})t^2 + (v_{x,u} - v_{x,v})t + p_{x,u} - p_{x,v})^2 + \\ &(\frac{1}{2}(a_{y,u} - a_{y,v})t^2 + (v_{y,u} - v_{y,v})t + p_{y,u} - p_{y,v})^2 + \\ &(\frac{1}{2}(a_{z,u} - a_{z,v})t^2 + (v_{z,u} - v_{z,v})t + p_{z,u} - p_{z,v})^2 + = r^2.\end{aligned} \quad (12)$$

For the equation above, only variable t is unknown, and t equals to the companion time between nodes v and u.

The companion time of two nodes can be expressed as the following when two nodes communicate with each other through multiple relay nodes,

$$t_{<v,u>} = \{\min t_{<x_i,x_{i+1}>} | x_1 = v, x_n = u, \\ dist(x_i, x_{i+1}) < r, \quad x_i \in V(t), \ i = 1, 2, \ldots, n-1\}. \tag{13}$$

The duration of a virtual edge is determined by the minimum companion time between the master node and other members, and therefore it can be expressed as

$$t_{dur,v} = \{\min t_{<v,u>} | v \text{ is master node}, \\ u \text{ is any slave node}\}. \tag{14}$$

In order to prevent members from being replaced frequently, the companion time between the master and a slave node must meet the following requirement.

$$t_{<v,u>} \geq \frac{r}{v_{max}}, \tag{15}$$

where v_{max} is the maximum allowable speed (speed limit) of the vehicle [34], which is related to traffic, vehicle, and road conditions.

1) COLLECTION OF VEHICLE INFORMATION

From the perspective of information exchange, there exist two types of approaches for collecting vehicle information: probing and beaconing. In the former one, the request messages are sent from the master node to other vehicles in the vicinity for collecting vehicle information into the BSM table. After the reception of a request message, each vehicle replies with a response message by including the required information. In the latter one, BSM with the vehicle information is broadcasted periodically in fixed intervals, and the BSM table is updated after the reception of a beacon message. Both approaches utilize the BSM table to calculate the time duration for a link between two vehicles, and then send requests to the vehicles for establishing virtual edge. The establishment of a virtual edge completes when the master node receives acknowledgments from all the vehicles. The whole process is implemented in simulation, and the comparison of the formation time of two approaches is illustrated in Fig. 5.

As seen from Fig. 5, with increasing of the number of slave nodes, the formation time will be increased. It is because that the time sending the request to establish virtual edge and the time returning ACKs will be increased. We also observe that the formation time of probing is longer than beaconing. In addition, according to the view of the author [32], the current setting in Table 1 does not affect the Quality-of-service (QoS) of networks. Therefore, we choose beaconing as the establishment approach for virtual edge in this paper.

2) AVAILABLE COMPUTATIONAL RESOURCE

To evaluate the computational resource of the virtual edge, which contain multiple slave nodes, "edge power" is introduced in this paper, and it is used to indicate the potential computational power provided by a virtual edge member. The edge power of node u is demonstrated as the following equation.

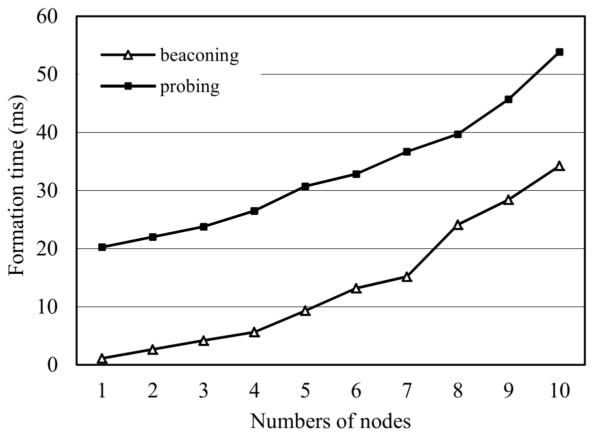

FIGURE 5. Formation time of a virtual edge for probing and beaconing.

$$P_u = \alpha_u f_u t_{u,v} \tag{16}$$

where f_u is CPU frequency, α_u is the ratio of idle resource, and $t_{u,v}$ is the companion time of node u, respectively. Then, the power of the virtual edge can be calculated as

$$P_v = \sum_{u \in U(t)} \alpha_u f_u t_{dur,v} \tag{17}$$

where $U(t)$ is the set of virtual edge members formed by node v at time t. The edge power increases with the number of member nodes and the duration of the virtual edge.

B. VIRTUAL EDGE FORMATION AND TASK OFFLOADING

The process of virtual edge formation and task offloading is illustrated in Algorithm 1. First, the basic vehicle information, including position and signal strength information, is achieved by exchanging BSM among neighbors. Based on this information, the edge power of an edge neighbor vehicle is calculated, and the neighbors are sorted by the edge power. After that, the neighbors with acceptable companion time are selected as the members of the virtual edge (slave nodes). In this way, both the computational capability and the relative mobility of vehicles are considered in the virtual edge formation. The selected members of the virtual edge are allocated with proper tasks according to the idle computational resources.

C. VIRTUAL EDGE MAINTENANCE

Due to the mobility of vehicles, the serving time of a virtual edge is constrained by the duration time. In order to execute long time computation tasks (i.e., large computation tasks that require longer processing time) smoothly, it is necessary to maintain the virtual edge. When a slave node leaves a virtual edge, the master node possibly needs to add a new node in order to guarantee the computational capability of the virtual edge. The master node also needs to trigger the maintenance

Algorithm 1 Processes for Virtual Edge Establishment and Computation Offloading

```
1: Each vehicle broadcasts BSM message.
2: Each vehicle updates the BSM table.
   (Master node executes the following steps.)
3: $V_v(t) \leftarrow \varnothing$;
4: Calculate duration of the neighbor vehicles according to
   Eq.(14) and update BSM table;
5: Calculate the edge power of all neighbors according to
   Eq.(16) and update BSM table;
6: Sort the neighbor nodes by the edge power with descending order;
7: for Each neighbor do
8:     if The companion time $>$ the predefined threshold then
9:         Add the neighbor into $V_v(t)$;
10:        Send a virtual edge request to vehicle $u$;
11:    else
12:        Break;
13:    end if
14: end for
15: Wait for a short time period;
16: for Each $u \in V_v(t)$ do
17:     if The ACK is received from the node then
18:         Offload block $d_u$ to the slave node $u$;
19:     else
20:         Wait for a short time period and try again for this node;
21:         Continue;
22:     end if
23: end for
```

process of the virtual edge in case of an unexpected link disconnection. However, there could be a failure in virtual edge maintenance due to the difficulty in finding appropriate slave nodes. If the virtual edge maintenance fails, the master node waits for a predefined time (i.e., 5 seconds by default) for the appearance of another node. This process continues until the virtual edge maintenance succeeds.

D. OPTIMAL NUMBER OF VIRTUAL EDGE MEMBERS

Being selected as a member of virtual edge needs to satisfy two conditions. The one is that the companion time between the master and the member vehicle is larger than the task execution time on the vehicle. The other one is that the sum of communication time and the execution time on the vehicle node is smaller than the execution time of the task on the master node. The maximum number satisfying two conditions above is the optimal number for a virtual edge.

With the increase of the number of slave nodes, the computational capability of virtual edge increases while the duration time decreases. There is a tradeoff between the computational capability and duration of virtual edge. The optimal value of the members of a virtual edge is dependent on the task size, network resources, vehicle velocity, and vehicle density. To obtain the optimal number of virtual edge, it is important to predict companion time and execution time on the local and remote nodes (on the slave nodes), respectively. However, obtaining the optimal number faces many challenges. First, the accuracy of the prediction to duration time and execution time influences the results of obtaining the optimal number. Second, the inter-vehicle distance and communication range also influence the optimal value. In this paper, we discuss the effect of different virtual edge members in various environments while the optimization algorithm for the number of virtual edge members is considered as future work. We discuss this in the future work part of the last section.

E. IMPLEMENTATION ISSUE

For the implementation aspect of the virtual edge, a container-based virtualization technology is introduced to enhance the efficiency of the resource management of vehicles. ‘‘Container’’ is a light-weight virtualization technology to achieve collaborative computing. For example, Docker [36] and Kubernetes [37] are containers that can be installed in vehicles to manage computational resources [30]. Compared to the traditional virtual machines (VMs), the containers have many advantages, such as fast start-up, resource sharing, and independent running environment [27], so that it is effectively utilized for task offloading and scheduling in VEC [38]. Container-based virtualization mainly depends on the sharing of OS kernel between containers, including hardware abstraction and device driver. Container-based virtualization not only improves computational resource efficiency but also avoids the interference of the resources by creating an isolated environment.

VI. EVALUATION

In order to validate the performance of the proposed virtual edge scheme, the discrete event simulator OMNeT++ and the mobility generator SUMO are integrated to provide a realistic wireless-enabled traffic network. We compare the proposed scheme with three benchmarks. The simulation process consists of three parts: the establishment of a virtual edge, virtual edge maintenance, and computation offloading. The parameters used in the simulation are listed in Table 1. A 4000m bi-directional straight road is used. In order to simulate the effect of intersections, we put 5 traffic lights evenly spaced among them along the road. The reference speed of the vehicle is limited to 50 km/h and the vehicle arrives at a random interval of [0.125, 0.33] vehicles per second. Each vehicle is installed with a container, so it is able to execute offloaded tasks from other vehicles. The master node starts to establish a virtual edge to offload the computation after the warm-up process is finished.

For the slave node selection, we compare the proposed scheme with three benchmarks, namely, the random approach, the minimum distance first approach, and the maximum available resources first approach. The detail of each benchmark is as follows.

TABLE 1. Parameters in the simulation.

Parameter	Value
Reference speed of vehicle	50 km/h
Vehicle arrival rate	uniform; [0.125, 0.33] vehicles/s
The number of vehicles	200
Antenna frequency	5.9 GHz
Bandwidth of V2V communications	10 MHz
Interval of sending BSM	1 second
Data rate of V2V communications	27 Mbps
Range of a single-hop communication	250 m
Computational resource at slave node	uniform; [0.1, 1] GHz
Warm-up time	200 seconds
Simulation time	320 seconds

- Random (rand) selects slave nodes among nodes within the single-hop communication range randomly.
- Minimum distance first (minDist) selects vehicles with comparatively shorter distance to the master vehicle as slave nodes.
- Maximum available resources first (maxRes) selects vehicles with comparatively higher available resources as slave nodes.
- Proposed scheme (Prop) considers the link duration between vehicles and the available computational resources jointly when selecting slave nodes.

A. VIRTUAL EDGE ESTABLISHMENT

Fig. 6 shows the duration of a virtual edge for different numbers of slave nodes (virtual edge members). As shown in Fig. 6, the duration of virtual edge in the proposed scheme is better than other approaches. There is a trend that the duration time of all four approaches becomes shorter with the increase of the number of slave nodes. The reason is that, involving more nodes in a virtual edge increases the risk of the breakage of a virtual edge. With the increase of the number of slave nodes, the available computational resource calculated by Eq. (16) increases too, as shown in Fig. 7. However, a very large virtual edge is impractical due to the mobility of the vehicles. Note that, the duration of the proposed scheme decreases dramatically when the number of slave nodes reaches six, or more than six. The reasons can be explained from the following two aspects. First, the number of vehicles is limited due to the limited single-hop communication range. Second, the stability of a virtual edge is influenced by the traffic light and the length of the lane. Thus, the optimal number of the members of a virtual edge depends on the traffic environment.

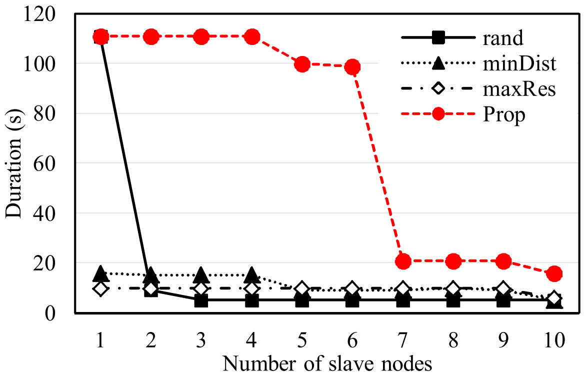

FIGURE 6. Duration time of virtual edge for different numbers of virtual edge members (without virtual edge maintenance).

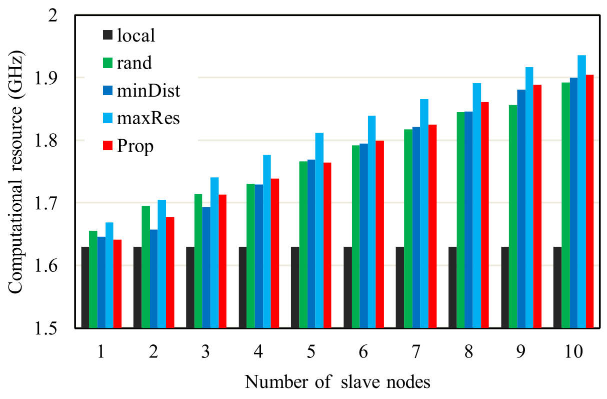

FIGURE 7. Computational resources for different numbers of virtual edge members (without virtual edge maintenance).

Fig. 8 shows the duration time of virtual edge for different vehicle arrival rates under various numbers of slave nodes. We observe an increase of duration time when the arrival rate increases from 0.1 to 0.2 vehicles per second. However, when the arrival rate is higher than 0.3 vehicles per second, the duration time decreases as the arrival rate increases. This is because the selection of virtual edge members considers both the connectivity (duration time) and the computational capability at each member. Since the computational capability of each vehicle is randomly generated in order to simulate a realistic vehicular environment in the simulation, some vehicles with shorter duration time and higher computational capability could be selected, resulting in a shorter duration time of virtual edge. Note that this is a correct behavior as the consideration of computation capability is also mandatory, especially for a task that requires a large amount of computational resources.

B. VIRTUAL EDGE MAINTENANCE

In order to measure the task offloading performance under the virtual edge maintenance, four metrics, namely, the duration of a virtual edge, unexpected link disconnection, failure of virtual edge maintenances, and available CPU cycle, are evaluated in this paper. The duration time of virtual edge for different numbers of slave nodes is illustrated in Fig. 9. We can see that the curve of the duration in the proposed scheme overlaps with the minDist benchmark and it is better than others. From the difference between Fig. 6 and Fig. 9, we can observe the importance of virtual edge maintenance. It is concluded from above that the virtual edge with maintenance

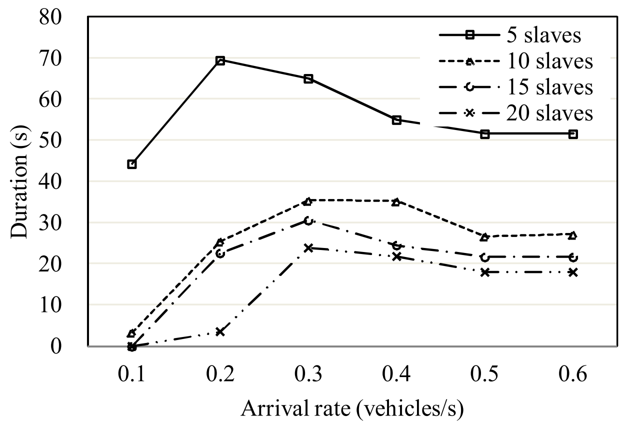

FIGURE 8. Duration time of virtual edge for different vehicle arrival rates under various numbers of slave nodes (without virtual edge maintenance).

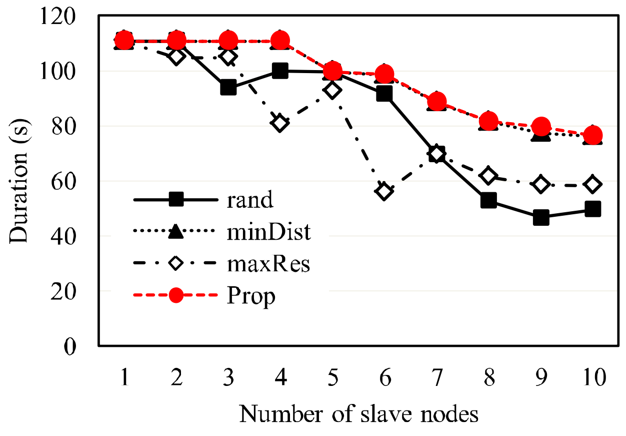

FIGURE 9. Duration time of virtual edge for different numbers of virtual edge members (with virtual edge maintenance).

is suitable for offloading long time computation tasks than other approaches.

An unexpected link disconnection of a slave node could result in a failure of returning the result of the computation to the master node. There are many reasons causing unexpected link disconnections, such as vehicle mobility and the weak signal quality. The number of unexpected link disconnections for different numbers of virtual edge members is illustrated in Fig. 10. It can be seen from the figure that, unexpected link disconnections increase with the increase of the number of slave nodes. The proposed scheme shows the best performance with the lowest number of link disconnections by considering the link duration for the virtual edge formation.

Fig. 11 illustrates the number of virtual edge maintenance failures for different numbers of virtual edge members. It is indicated that the number of failures of the proposed scheme equals to the minDist benchmark, and it is smaller than other approaches. This is because the proposed scheme reduces the possibility of virtual edge failures by considering vehicle mobility, which results in a reduction of virtual edge maintenance failures as well.

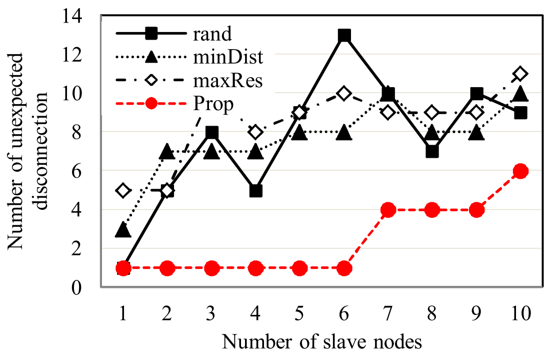

FIGURE 10. The number of unexpected link disconnections for different numbers of virtual edge members.

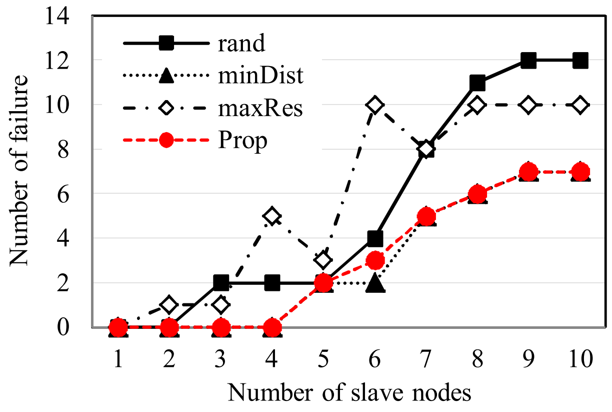

FIGURE 11. The number of virtual edge maintenance failures for different numbers of virtual edge members.

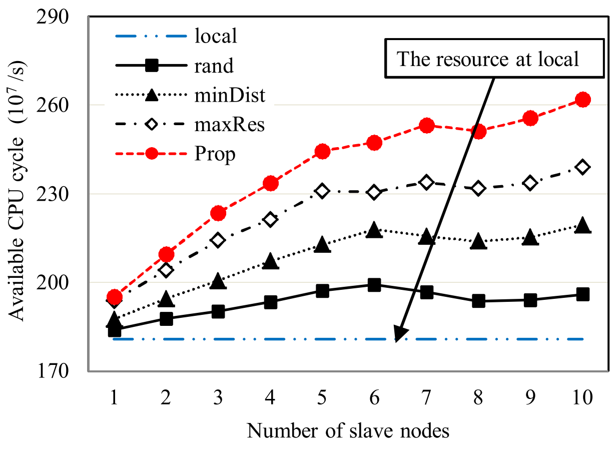

FIGURE 12. Available CPU cycle of virtual edge for different numbers of slave nodes.

We observe from Fig. 9 and Fig. 11 that for some cases, the proposed scheme and minDist show the same values. This does not mean the proposed scheme and minDist select the same virtual edge members. From the result of minDist, we can observe that the selection of virtual edge member by considering solely the inter-vehicle distance can achieve an acceptable duration time and number of failures. However, as we will show in Fig. 13 and Fig. 14, the minDist shows a

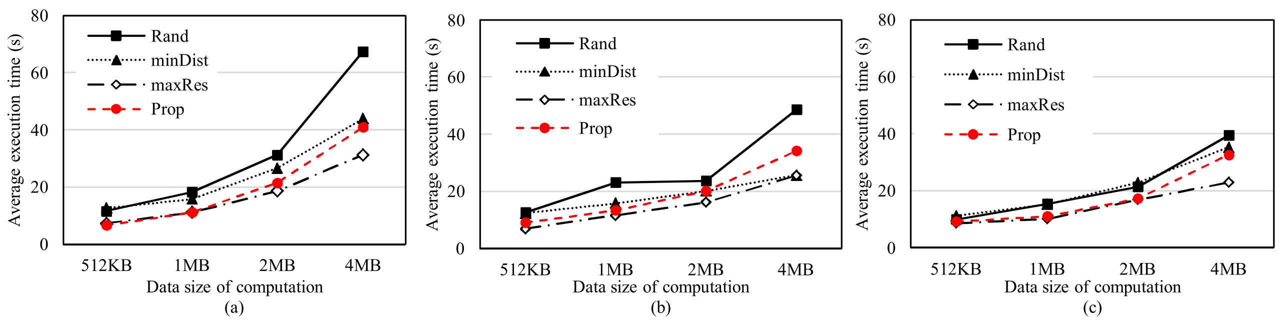

FIGURE 13. **Impact of data size on the average task execution time: (a) with 2 slave nodes; (b) with 3 slave nodes; (c) with 4 slave nodes.**

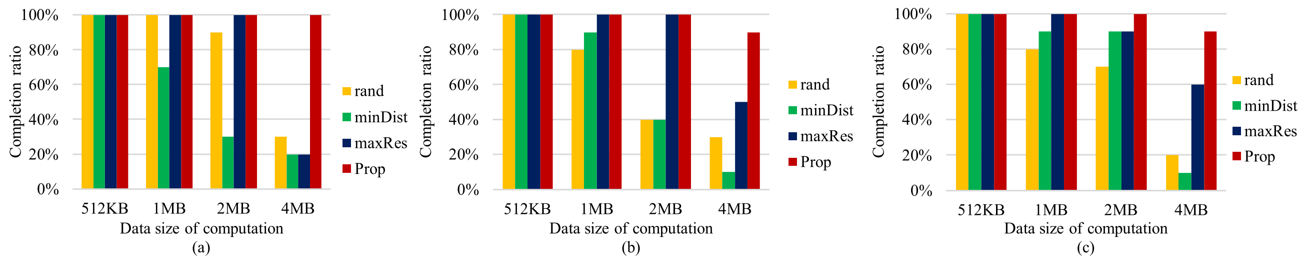

FIGURE 14. **Impact of data size on the task completion ratio: (a) with 2 slave nodes; (b) with 3 slave nodes; (c) with 4 slave nodes.**

poor performance in terms of the average execution time and completion ratio.

The power of a virtual edge is calculated by Eq. (16), which are the available CPU cycles at all the members of the virtual edge throughout its lifetime. Fig. 12 illustrates the comparison of virtual edge power per second (i.e., the available CPU cycles at all the members per second) for different numbers of slave nodes. From this figure, we find the following results. First, the power of a virtual edge is stronger than a single vehicle (the local resource as indicated in the figure). Second, the proposed scheme is better than the other approaches due to the joint consideration of the virtual edge duration and available resources. Third, with the increase of the number of slave nodes in a virtual edge, the power of a virtual edge improves. By comparing Fig. 7 and Fig. 12, we find that the power of a virtual edge with maintenance is stronger than a virtual edge without maintenance.

Containing more slave nodes in a virtual edge also faces some challenges. For example, the prediction of the virtual edge duration becomes more difficult, and the probability of the virtual edge maintenance increases, as shown in Fig. 10 and Fig. 11. With increase of the number of slave nodes, the prediction of the virtual edge duration becomes more complex. Additionally, adding more members can inevitably lead to a vehicle with weak performance becoming one of the members. An increase of the number of slave nodes leads to a higher probability of link failure between the master node and a member node. As a result, the number of unexpected link disconnections increases, and the number of virtual edge maintenances increases. Unexpected link disconnections during the execution also have negative effects on collaborative offloading tasks. It can be seen from Fig. 10, a virtual edge with six slave nodes obtains the optimal computational resources since the number of unexpected link disconnections is the lowest. A similar result can be seen from Fig. 11, whereby a virtual edge with four slave nodes obtains the largest virtual edge power due to the fact the number of virtual edge maintenances is the smallest in that case.

C. COMPUTATION OFFLOADING

In order to measure the performance of a virtual edge for computation offloading, we conduct a simulation for offloading computation tasks from the master node to the slave nodes. To eliminate the influence of the duration time of a virtual edge, we choose scenarios with 2, 3, and 4 member nodes, respectively, and evaluate the impact of the data size on the average execution time and completion ratio. Fig. 13 shows the impact of data size on the task completion ratio. While the existing baselines show a decrease in the completion ratio with the increase of task load, the proposed scheme achieves a significant advantage over them. This is because the proposed scheme considers the computational capability of vehicles in the virtual edge formation. We can also observe in Fig. 14 that with the increase of data size and the number of slave nodes, the task completion ratio of all schemes drops. As compared with other three baselines, the proposed scheme significantly improves the completion ratio by considering the duration time of communication links between nodes, and reducing the number of link disconnections in the virtual edge.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose virtual edge, which is a collaborative task offloading scheme that utilizes sporadic computational resources on vehicles efficiently for task execution in vehicular environments. The proposed scheme is able to generate an efficient virtual edge by considering the vehicle mobility and computational capability of vehicles jointly in the virtual edge formation. We use extensive computer simulations to evaluate the performance of the proposed scheme by comparing its performance with existing baselines. Simulation results show that the proposed scheme can generate efficient virtual edge, and therefore it can achieve a much higher task completion ratio with reasonable average task execution time as compared with existing baselines.

In this paper, we discussed the effect of different virtual edge members in various environments. In future work, we will discuss how to optimize the number of virtual edge members in a dynamic vehicular environment according to varying application requirements. We will consider using some AI-empowered approaches to further improve the performance of our scheme by enhancing the virtual edge selection process. We will also evaluate the performance of our scheme in different applications by conducting real-world experiments.

REFERENCES

[1] N. Hassan, K.-L.-A. Yau, and C. Wu, "Edge computing in 5G: A review," *IEEE Access*, vol. 7, pp. 127276–127289, 2019.

[2] C. Huang, R. Lu, and K.-K.-R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[3] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.

[4] N. Cha, C. Wu, T. Yoshinaga, and Y. Ji, "Virtual edge: Collaborative computation offloading in VANETs," in *Proc. 10th Int. Conf. EAI MONAMI*, Nov. 2020, pp. 79–93.

[5] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[6] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[7] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[8] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[11] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[12] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, Feb. 2019.

[13] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.

[14] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Feb. 2019.

[15] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 48–54, Aug. 2018.

[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[17] C. Wu, Z. Liu, D. Zhang, T. Yoshinaga, and Y. Ji, "Spatial intelligence toward trustworthy vehicular IoT," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 22–27, Oct. 2018.

[18] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task replication for vehicular edge computing: A combinatorial multi-armed bandit based approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates Dec. 2018, pp. 1–7.

[19] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[20] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[21] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.

[22] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Collaborative vehicular edge computing networks: Architecture design and research challenges," *IEEE Access*, vol. 7, pp. 178942–178952, 2019.

[23] S. Buda, S. Guleng, C. Wu, J. Zhang, K.-L.-A. Yau, and Y. Ji, "Collaborative vehicular edge computing towards greener ITS," *IEEE Access*, vol. 8, pp. 63935–63944, 2020.

[24] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, early access, Mar. 18, 2020, doi: 10.1109/TITS.2020.2980422.

[25] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 4, pp. 1155–1165, Dec. 2020.

[26] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep. 2018.

[27] R. Morabito, J. Kjallman, and M. Komu, "Hypervisors vs. Lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf. Cloud Eng.*, Tempe, AZ, USA, Mar. 2015, pp. 386–393.

[28] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of microservices for IoT using docker and edge computing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 118–123, Sep. 2018.

[29] J. Tang, R. Yu, S. Liu, and J.-L. Gaudiot, "A container based edge offloading framework for autonomous driving," *IEEE Access*, vol. 8, pp. 33713–33726, 2020.

[30] X. Huang, R. Yu, S. Xie, and Y. Zhang, "Task-container matching game for computation offloading in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, early access, May 8, 2020, doi: 10.1109/TITS.2020.2990462.

[31] N. Akhtar, S. C. Ergen, and O. Ozkasap, "Vehicle mobility and communication channel models for realistic and efficient highway VANET simulation," *IEEE Trans. Veh. Technol.*, vol. 64, no. 1, pp. 248–262, Jan. 2015.

[32] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the performance of IEEE 802.11p and LTE-V2 V for the cooperative awareness of connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10419–10432, Nov. 2017.

[33] *WAVE Multi-Channel Operation*, Standard 1609.4-2006, 2006.

[34] B. Khondaker and L. Kattan, "Variable speed limit: A microscopic analysis in a connected vehicle environment," *Transp. Res. C, Emerg. Technol.*, vol. 58, pp. 146–159, Sep. 2015.

[35] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar. 2019.

[36] *Docker*. Accessed: Oct. 15, 2020. [Online]. Available: https://www.docker.io/

[37] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[38] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018.

NARISU CHA is currently pursuing the Ph.D. degree with the Graduate School of Informatics and Engineering, The University of Electro-Communications (UEC). His current research interests include mobile edge computing, vehicular ad hoc networks, and resource management.

CELIMUGE WU (Senior Member, IEEE) received the M.E. degree from the Beijing Institute of Technology, China, in 2006, and the Ph.D. degree from The University of Electro-Communications, Japan, in 2010. He is currently an Associate Professor with the Graduate School of Informatics and Engineering, The University of Electro-Communications. His research interests include vehicular networks, edge computing, the IoT, intelligent transport systems, and application of machine learning in wireless networking and computing. He received the IEEE Computer Society 2019 Best Paper Award Runner-Up. He is the Chair of the IEEE TCGCC Special Interest Group on Green Internet of Vehicles and the IEEE TCBD Special Interest Group on Big Data with Computational Intelligence. He serves as an Associate Editor for IEEE Open Journal of the Computer Society, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Green Communications and Networking, and IEEE Access.

TSUTOMU YOSHINAGA (Member, IEEE) received the B.E., M.E., and D.E. degrees from Utsunomiya University, in 1986, 1988, and 1997, respectively. From 1988 to July 2000, he was a Research Associate with the Faculty of Engineering, Utsunomiya University. From 1997 to 1998, he was also a Visiting Researcher with the Laboratory of Electro-Technical. Since August 2000, he has been with the Graduate School of Information Systems, The University of Electro-Communications, where he is currently a Professor. His research interests include computer architecture, interconnection networks, and network computing. He is a member of ACM and IPSJ and a Fellow of IEICE.

YUSHENG JI (Senior Member, IEEE) received the B.E., M.E., and D.E. degrees in electrical engineering from The University of Tokyo. In 1990, she joined the National Center for Science Information Systems (NACSIS), Japan. She is currently a Professor with the National Institute of Informatics (NII) and The Graduate University for Advanced Studies, SOKENDAI. Her research interests include network architecture, resource management, and quality of service provisioning in wired and wireless communication networks. She is/has been a Symposium Co-Chair of IEEE GLOBECOM 2012, 2014, and a Track Co-Chair of IEEE VTC2016-Fall and VTC2017-Fall. She is/has been an Editor of IEEE Transactions on Vehicular Technology.

KOK-LIM ALVIN YAU (Senior Member, IEEE) received the B.Eng. degree (Hons.) in electrical and electronics engineering from Universiti Teknologi PETRONAS, Malaysia, in 2005, the M.Sc. degree in electrical engineering from the National University of Singapore, in 2007, and the Ph.D. degree in network engineering from the Victoria University of Wellington, New Zealand, in 2010. He is currently a Professor with the Department of Computing and Information Systems, Sunway University. He is also a Researcher, a Lecturer, and a Consultant in cognitive radio, wireless networks, applied artificial intelligence, applied deep learning, and reinforcement learning. He serves as a TPC Member and a Reviewer for major international conferences, including ICC, VTC, LCN, GLOBECOM, and AINA. He was a recipient of the 2007 Professional Engineer Board of Singapore Gold Medal for being the Best Graduate of the M.Sc. degree, in 2006 and 2007. He also served as the Vice General Co-Chair for ICOIN'18, the Co-Chair for IET ICFCNA'14, and the Co-Chair (Organizing Committee) for IET ICWCA'12. He serves as an Editor for the *KSII Transactions on Internet and Information Systems*, an Associate Editor for IEEE Access, a Guest Editor for the Special Issues of IEEE Access, *IET Networks*, the *IEEE Computational Intelligence Magazine*, *Journal of Ambient Intelligence and Humanized Computing* (Springer), and a Regular Reviewer for more than 20 journals, including the IEEE journals and magazines, the *Ad Hoc Networks*, and the *IET Communications*.