A Proactive Reliable Mechanism-Based Vehicular Fog Computing Network

Luobing Dong, Qiufen Ni, Weili Wu, *Senior Member, IEEE*, Chuanhe Huang, *Member, IEEE*, Taieb Znati, *Member, IEEE*, and Ding Zhu Du, *Member, IEEE*

***Abstract*—As vehicles are becoming more and more intelligent, mobile data traffic in vehicular *ad hoc* network (VANET) has been increasing dramatically. This makes the communication capacity of VANET systems and the computing resources of vehicles insufficient. In the meantime, location-aware large-scale distributed services with very low latency and high reliability are demanded by most of the novel functions, such as accident alarming, and congestion warning, in the intelligent transportation system. To meet these claimed characteristics of VANET, we first present a novel architecture that integrates vehicular fog computing and vehicle-to-vehicle (V2V) communication technologies. Lower latency and higher quality services can be supplied to vehicles by nearby fog servers, which are virtualized from vehicles that locate close enough and communicate using the V2V link. However, like all collaborative systems, computing reliability is vital to collaborative VANET. In this article, we design a novel energy-efficient proactive replication mechanism. Follower vehicles calculate with a lazy rate act as backups of host vehicles to ensure the reliability of the system. Considering the time sensitivity of computing requirements in VANET, the upper bound on the total number of failures is proposed through theoretical analysis. Then, the lower bound on the lazy calculating rate of followers is derived by balancing the tradeoffs between delay and energy. A fast algorithm for searching this lower bound based on the discrete Newton method is also proposed. Results of numerical experiments show that our new mechanism is effective in energy saving and reliability enhancing.**

***Index Terms*—Fog computing, discrete Newton method, reliability, vehicle-to-vehicle (V2V), vehicular *ad hoc* network (VANET).**

I. Introduction

VEHICULAR *ad hoc* network (VANET) is an important part of the intelligent transportation system (ITS), which can support many applications, such as information dissemination, accident alarming, congestion warning, navigation, entertainment, and so on. It brings many conveniences to people's lives and makes traffic safer [1]. VANETs are expected to accommodate a large number of data-heavy mobile devices and multiapplication services. However, as the number of vehicles increases rapidly, the mobile data traffic in VANET is also increasing explosively. This large amount of data causes the insufficiency of communication capacity and computing resources. In order to solve these resource shortages, some technologies emerge, such as vehicle-to-vehicle (V2V) communication and fog computing.

To solve the novel challenging needs of modern transportation systems, such as very low latency and location awareness, VANETs cannot heavily depend on the remote resources of cloud servers [2]. Because the traditional cloud computing brings intolerant delay and latency to VANETs. As an example, the modern smart city system wants to prevent traffic accidents from some unforeseen circumstances, such as landslides and icy pavement, by analyzing the road surrounding pictures that are collected by vehicles. However, sometimes accidents just happen so quickly that pictures have not even been transmitted to the cloud server from vehicles. Edge computing and fog computing can minimize delay and latency by moving cloud computing capabilities closer to the vehicles. This makes them two of the natural solutions of VANETs [2]–[4]. Unlike edge computing that deploys servers on special nodes such as base stations (BSs) network controller/macro BS, fog computing can place its fog servers everywhere [4]. Especially, vehicles can also serve as fog servers. In the above example, if scene photographs can be analyzed locally by the collaboration of surrounding vehicles, there is still time to avoid accidents. Fog computing couples with high-speed transmission provided by the V2V link can just make this kind of local vehicular collaborative computing possible.

Infrastructures such as roadside units (RSUs) are always used as fog servers to serve nearby vehicles [5]. But adding and upgrading infrastructures in VANET will inevitably lead to high business costs. In fact, fog service can be obtained in a cheaper way. As we all know, not all vehicles in VANET have the same configuration. This makes the computing capabilities of vehicles different from each other. In general, low-end vehicles have poor computing resources. High-end vehicles have richer computing resources which are always redundant with respect to their normal demand. Although low-end

Manuscript received May 5, 2020; revised June 7, 2020; accepted June 30, 2020. Date of publication July 7, 2020; date of current version December 11, 2020. This work was supported in part by the National Science Foundation under Grant 1747818 and Grant 1907472; in part by the Fundamental Research Funds for Central Universities under Grant JB161004; in part by the Department of Energy under Contract DE-SC0014376; and in part by the National Science Foundation of China under Grant 61772385 and Grant 61572370. *(Corresponding author: Luobing Dong.)*

Luobing Dong is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China (e-mail: lbdong@xidian.edu.cn).

Qiufen Ni and Chuanhe Huang are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: niqiufen@whu.edu.cn; huangch@whu.edu.cn).

Weili Wu and Ding Zhu Du are with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: weiliwu@utdallas.edu; dzdu@utdallas.edu).

Taieb Znati is with the Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: znati@cs.pitt.edu).

Digital Object Identifier 10.1109/JIOT.2020.3007608

vehicles are equipped with computing resources that are sufficient to handle normal conditions, they may also get some complex requirements that exceed their capabilities. To complete these complex requirements, low-end vehicles may need help. Without additional investment, high-end vehicles could be more ideal helpers than RSUs. Since vehicles manage their own storage and computational resources, virtualization technology enables them to act as fog servers for nearby vehicles. This proximity-service requirement in this mobile environment requires more advanced communication technologies. V2V communication technology enables vehicles to communicate directly without going through the BS [6]. It removes the connection delay and has larger communication capacity and longer communication distance. Furthermore, as the dominant communication technology between mobile terminals in the 5G system, V2V will also make a significant decrease in the number of RSUs. These properties of V2V enable vehicles to supply efficient fog services to other nearby vehicles.

In this article, we present a novel VANET architecture which integrates V2V communication technology into vehicular fog computing network. This architecture is termed as V2V-enabled vehicular fog computing architecture (VVFCA). By identifying the amount of idle computing resources of surrounding vehicles, vehicles choose some near-located helpers to complete their computing tasks together. All vehicles which are working on one same task form a virtual cluster. This collaborative model can efficiently enhance a vehicle's ability to handle complex requirements with low latency.

However, as with all collaborative systems, the reliability of collaboration among different vehicles is vital to collaborative VANET (especially for VVFCA-based networks) [7], [8]. Any computational failure in any vehicle will inevitably lower the computational efficiency of the whole task. In addition, failures in VANET can lead to more serious consequences, such as traffic jams and traffic accidents.

The reliability issue of collaborative VANET mainly comes from the continuous changing of the relative location of vehicles. This change will cause some vehicles to be no longer able to collaborate with some other vehicles which are in the same cluster. For example, one high-end vehicle is helping another low-end vehicle to complete a computing task. But the high-end vehicle must exit the highway immediately. The leave of this high-end vehicle will lead to the abort of the ongoing task. In fact, the change of vehicles' relative positions that may disrupt some existing collaborative relationships always occurs at the end of a travel phase of one vehicle. Most vehicles always know their leave ahead of time. If we can take some measures in this interval, the reliability of the system could be improved.

Different from passive maintenance and detection of the stability of collaborative clusters [9]–[11], we design a proactive collaborator replication mechanism. Each vehicle has one lazy shadowing follower for its collaborative computing tasks. This shadowing follower executes the same task at a reduced rate. When one vehicle wants to leave the cluster, it will proactively notify its shadowing follower to speed up its calculation. This proactive replication mechanism is energy efficient and can improve the computing reliability of VVFCA-based networks.

The key contributions of this article are summarized as follows.

1) We define a new paradigm of V2V-enabled vehicular fog computing networks. By virtualizing vehicles as fog servers and transferring data through V2V links among vehicles, this new architecture can effectively improve the communication capacity and computing capacity of VANET.
2) We address the reliability issue of VANET and design a novel energy-efficient proactive replication mechanism. Vehicles solve the computational reliability problem by themselves instead of relying on the central nodes of clusters. This distributed construction improves the robustness of the system.
3) We first propose the difference operator for the progress difference between two failures and the upper bound on the tolerant total number of failures. Based on these two parameters, the expected completion time of the VVFCA-based system is quantified.
4) The lower bound on the follower's lazy calculating rate is given by balancing the tradeoffs between delay and energy. We innovatively propose a fast algorithm for this lower bound searching based on the discrete Newton method.

The remainder of this article is organized as follows. Section II discusses the related work. We construct the new architecture in Section III. The proactive reliable mechanism is also designed in this section. In Section IV, we give the theoretical analysis of the proposed reliable mechanism. The lower bound of the lazy following rate is proposed by using the discrete Newton method. Section V presents the simulation results, and finally, the conclusion is presented in Section VI.

II. Related Work

In this section, we conclude some existing works related to our study from three perspectives: 1) fog computing in VANET; 2) D2D in VANET; and 3) discrete Newton method.

A. Fog Computing in VANET

Many works have already studied considering vehicles as fog computing infrastructures, especially integrating vehicular fog computing with software-defined networking (SDN) technology [12]–[17]. These methods can improve the QoS of vehicular applications and protocols. Because it offloads the computing services from the cloud to the edge of networks and makes network control decisions locally. Park and Yoo [18] believed that the mobility of vehicles leads to unstable links between vehicles. They introduce fog computing and SDN technology into VANET and classify mobile information into three categories. In order to help the controller supervise network resources, they measure the signal strength of the link between the controller and each switch. They present an intelligent maintenance method combining with network information and give some applications in VANET. Tang *et al.* [19] proposed a parking slot allocation strategy and a four-layer architecture. They introduce fog computing into VANET to provide real-time parking slot information. They

comprehensively consider the factor that affects decision making and give drivers options from which to choose their own preferences.

There are also some researches on the reliability issue in fog computing-based VANET. Xiao *et al.* [20] studied the situation that potential processing nodes may have failures. They use a task allocation strategy to increase the reliability of VANET and also introduce delay constraint and retransmission mechanism into hybrid SDN/fog computing VANET system to enhance the real-time characteristics of VANET. Pereira *et al.* [2] proposed a proof-of-concept mechanism to perform data analysis in hybrid VANET/fog architecture. It can detect potential city traffic anomalies and then deliver bus stop arrival time to end devices. This method enables the development of reliable smart mobility applications.

These above researches prove that fog computing is completely feasible for VANETs. Some basic mechanisms, such as QoS guarantee and real-time services have been well implemented. We intend to take the matter further and try to virtualize the vehicle as a fog server to make the system more efficient. Although researchers have conducted researches on the reliability issue in fog computing-based VANETs, their solutions are mainly to passively maintain the stability of the vehicle cluster. We will propose a proactive guarantee scheme for the first time.

B. D2D in VANET

When both communication parties are vehicles, D2D can be named as V2V. Researchers always use these two concepts vaguely in VANETs. Therefore, we will comprehensively review D2D in VANET here. Min *et al.* [21] studied the reliability in D2D communication networks. They propose an interference retransmission method to improve the reliability of the D2D receiver. The channel assignment problem in cellular-VANET heterogeneous wireless networks has been studied in some research. Feng *et al.* [22] studied the resource allocation problem to maximize the overall network throughput while guaranteeing the QoS requirements for both D2D users and regular cellular users (CUs). To avoid interference in D2D communication, the authors develop a three-step scheme. It investigates power control for each D2D pair and its possible CU partners to maximize overall throughput and determines a specific CU partner for each admissible D2D pair. In the VANET scenario, the problem of maximizing overall throughput has attracted attention. Wei *et al.* designed a resource allocation strategy to jointly address the spectrum allocation, power controlling, and spectrum sharing. Their objective is to maximize the total throughput of CU's equipment and nonsafety vehicular user's equipment. They also propose a three-step decomposing scheme, which is nested with the interior point method and the matching algorithm. Meng *et al.* [23] considered the problem of resource reallocation and proposed a time dynamic problem. Their objective is to constrain the network reallocation rate and increase the number of guaranteed services at a low cost in a resource-limited vehicular network. There are also some studies on D2D discovery in VANET. Chour *et al.* [24] focused on how to effectively reduce network load in the VANET. They offload some D2D discovery traffic and process D2D sessions involving vehicular users. The inherent information of the RSU can be used for the proposed scheme. An analytical model is developed to analyze the duration of D2D discovery in a highway scenario.

From the above researches, we know that most V2V communication problems, such as reliability, resource allocation, and interference, in VANETs have been studied. This means low cost and reliable V2V communication in VANETs is feasible. We will use this as a basis to achieve low cost and reliable collaborative computing among vehicles by integrating V2V and fog computing.

C. Discrete Newton Method

Newton method is a classic and powerful method in continuous nonlinear optimization. In 1993, Radzik [25] proposed a discrete Newton method for the linear fractional combinatorial optimization problem, the core of which is to search the unique root of a piecewise linear decreasing function. The number of searching iterations is polynomial in the input size but independent of the input numbers. They also provide a strong polynomial-time algorithm. The discrete Newton method is used to solve the discrete nonlinear system generated by each step after discretization [26]. The authors also analyzed the convergence of their proposed scheme. The work in [27] points out that Newton's method is a popular numerical method for the solution of the equilibrium of equations. The numerical procedure does not converge within a certain neighborhood of the critical state of equilibrium. Riks [27] extended the stability analysis beyond the stability limit.

The discrete Newton method gives a fast way to find the unique root of a piecewise linear decreasing function. In Section IV, a fast searching algorithm for the lower bound of followers' lazy computing rate is designed based on the discrete Newton method.

III. V2V-Enabled Vehicular Fog Computing Network

In this section, we propose a novel architecture to supply fog services using moving vehicles in VANET based on V2V communication. First, with V2V, communication between two adjacent vehicles can be enhanced. Moving vehicles can communicate directly with each other to exchange data, computing requirements, and so on. Second, vehicles that are working for the same computing task form a virtual cluster. The host vehicle is responsible for task allocation and cluster management. Third, each vehicle in the cluster chooses one following replica (we call it as a follower) from nearby vehicles for its computing task which is calculated for the host vehicle. The follower calculates at a very low rate. When one vehicle wants to leave the cluster, its follower will speed up computation and takes the place of the leaving vehicle as a new helper for the host vehicle. The task scheduler of this follower is in charge of the detailed implementation of speeding up computation. When it receives the leave notification, the follower will send the speeding up requirement to its task scheduler. There are many

methods to do this, such as promoting priority as highest and increasing CPU usage time.

A. Vehicles as Fog Servers

As an established paradigm, the cloud computing system makes resources, such as storage, applications, and computing, available to customers. Customers can acquire on-demand resources based on a pay-per-usage basis. Because of its theoretically infinite virtual resources, the cloud computing system used to be widely accepted as the central computational backbone of most mobile networks including VANET. But with the increase of vehicles, location-aware large-scale distributed services with very low and predictable latency are demanded. Due to the high latency of long-distance transmission, lack of location awareness, and missing geographically distributed data centers close to vehicles in VANET, cloud computing is not able to meet the claimed characteristics of VANET. As mentioned above, many vehicles, especially high-end vehicles, have redundant computing resources. Because vehicles manage their own storage and computational resources, it is a promising approach to sharing these resources and use them to supply close fog service.

Fig. 1 shows our solution of using vehicles as fog servers. Virtualization technology is used to create isolated virtual resources from the physical hardware of vehicles. These virtual resources are dynamically shared as fog computing services among vehicles. At the *lowest level* of the stack in Fig. 1, there are physical resources that belong to a physical host vehicle. Physical resources represent the collection of CPU cores, memory, storage, and so on. The host operating system (OS) in the *second-level* manages these physical resources and transforms the program of higher levels into instructions that can be directly executed by the machine. The host OS is the original factory supplied. The original factory-supplied applications will run as normal. The *virtualization layer* deals with network, storage, and computation virtualization. Through virtualization, resources of one vehicle are assigned to clients dynamically on-demand in the form of *fog services/applications*. Each application has a corresponding service. In a virtualization environment, different fog services/applications are isolated from each other in their respective virtual partition.

A variety of conventional complex applications, such as long content delivery, image recognition, and road condition recognition, are preinstalled as fog applications in each vehicle. When one vehicle runs some of these applications and its own resources are short, it will apply for assistance from nearby vehicles. These assistances will be provided in the form of fog services by nearby vehicles that are preinstalled corresponding fog services. The detailed process will be described in the following sections.

There are many reasons why high-end vehicles agree to share their computing resources with other vehicles, such as making money and earning more opportunities to borrow resources from high-end vehicles. Service incentive is a hot research topic in the field of cloud computing. There are many mature mechanisms [28].

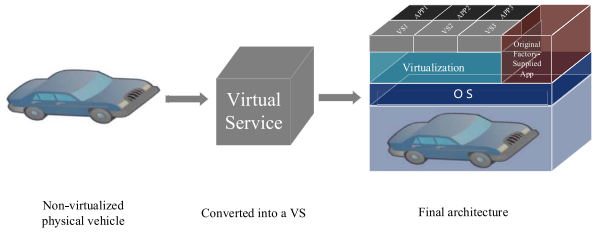

Fig. 1. Hardware virtualization of fog computing service.

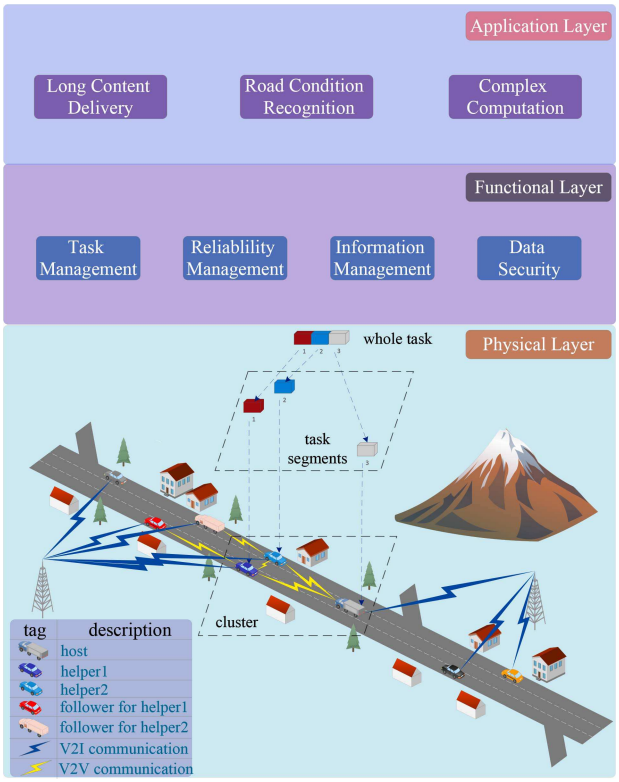

Fig. 2. Architecture for the proposed framework.

B. V2V-Enabled Vehicular Fog Computing Network

The network based on VVFCA aims to improve the computational ability of each vehicle by using fog services which are supplied by nearby vehicles. It provides a reliable distributed computing environment. Complex computing tasks of one vehicle will be split into segments such that collaborative computing can be provided by nearby vehicles. The task initiator is responsible for the allocation and maintenance of all task fragments. BSs are responsible for collecting the location and redundant resource information of vehicles and broadcasting this information to vehicles. Fig. 2 shows the new architecture. It is divided into three layers: 1) physical layer; 2) functional layer; and 3) application layer.

Physical Layer: A set of moving vehicles connect to the wireless network via a wireless interface, such as a wireless fidelity access point (WiFi AP) and BS. Vehicles report their information, including location, free resources, and willingness as a helper to its connected WiFi AP or BS periodically. WiFi AP or BS will broadcast these information. Vehicles usually execute tasks by themselves. However, they may sometimes

get some complex requirements that are beyond their own capabilities. At this time, they will choose some vehicles as their helpers from nearby available fog servers based on the information they got from the broadcast from BSs or WiFi APs. Here, available fog servers represent vehicles that can meet all the following conditions: have redundant resources, wish to be a helper, supply the corresponding virtual service, and within the communication range of V2V. One important point is that the communication range of V2V refers to the maximum communication distance between two vehicles, which depends on the wireless technology used in the network, such as 5G and WiFi. We call the help seeker as the host vehicle for the task. It will split the task into multiple segments and allocate them to the helpers. The host vehicle and all of its helpers form a virtual cluster through V2V links. The host vehicle maintains the cluster and collects the execution results of task segments. The cluster will be disbanded after all results of task segments have been reported to the host vehicle.

Functional Layer: It consists of four modules: 1) task management; 2) reliability management; 3) information management; and 4) data security. The task management module is responsible for task splitting, task allocation, and results in consolidation. The virtual cluster is also managed by this module. Due to the uncertainty of vehicle movement, the reliability of cooperation between vehicles must be considered. As already mentioned, we design a proactive reliable replication mechanism (PRRM) for the system and equip it into the reliability management module. Vehicles submit their information periodically. BS broadcasts all information it has through the information management module periodically. The data security module secures data and protects vehicles' privacy by encrypting sensitive data in the fog system and ensuring compliance with applicable regulations.

Application Layer: The system supplies all applications and their corresponding fog services in this layer. Host vehicles run applications and call corresponding fog services which are run in their helpers.

C. Proactive Reliable Replication Mechanism

In VANETs, reliability is an important criterion for most applications. It is mainly derived from realistic nonuniform distributions of vehicles and dynamically changed relative locations among vehicles which are caused by the random vehicular mobility. For the VVFCA-based networks, it affects the reliability of the virtual clusters and finally affects the reliability of computing for which the clusters are working. To solve this problem, the conventional method is to create stable enough clusters. But this goal is very hard to achieve. Because vehicles tend to move between points of interest that are often changed in time. Different from the conventional methods, we assume that the virtual cluster is not stable and design a novel proactive replication mechanism for the reliability of collaborative computation in the VVFCA-based network.

In VVFCA-based networks, there are two main issues that will affect the computation reliability of the system: 1) computation failure and 2) vehicle leave. We will give the definition of these two issues before designing the corresponding solutions.

Definition 1 (Computation Failure): When the execution of the current task reaches a point from where it cannot go any further due to some internal issues with its host, then this is called computation failure.

A computation failure could occur at a host, helper, or follower in VVFCA-based networks. Reasons for a computation failure could be any internal error, such as disk failure, system crash, and so on. When a failure occurs at a host or helper, its follower will fill its place. We will give the details later. For simplify, when a failure occurs at a follower, we just remove this follower and choose a new follower for its followee. If there is no available candidate follower and its followee hits a failure, the task will be halt.

Definition 2 (Vehicle Leave): When one helper or follower cannot continue to serve its host or followee, then this is called vehicle leave.

A vehicle leave can be either a follower leave or a helper leave. In essence, one vehicle leave refers to that a vehicle jumps out of the available V2V communication range of its partner (host or followee) and can no longer supply collaborative computation. The reason may be changing route, no catching up, and so on. For a helper leave, its follower will prop the task up if the follower is in the available communication range of the host. If the follower is not in the available communication range of the host, the current task will be terminated. We will give the details later. If a follower leaves, we just choose a new follower for its followee.

We will describe the details of the PRRM as follows.

We assume that vehicle v_i with maximum calculation speed s_i gets a complex application requirement $\mathscr{R}_i$ of size r_i. This requirement has to be completed in $t^i_{\max} \ll (r_i/s_i)$. Obviously, this vehicle must ask for help from nearby fog servers. We call this vehicle v_i as the host vehicle for $\mathscr{R}_i$. The host vehicle can choose $n-1$ closest fog servers to help it in $m \geq n-1$ available nearby fog servers. We use vector $\mathscr{V}_i = (v_i^0, v_i^1, \ldots, v_i^{n-1})$ to represent the host vehicle and all its helpers. The original requirement is split into a set of n task segments $\mathscr{F}_i = (f_i^0, f_i^1, \ldots, f_i^{n-1})$. For simplicity of discussion, we assume that task segment f_i^j is allocated to helper v_i^j and task f_i^0 is allocated to v_i. We use v_i^0 to represent v_i. The maximum calculation speed of v_i^j is s_i^j. v_i and all its helpers consist of the virtual cluster C_i for $\mathscr{R}_i$. For the reliability of the calculation of each task segment f_i^j, we set one shadowing replication for v_i^j in another vehicle $v_i^{j'}$. We call $v_i^{j'}$ as the follower of v_i^j. $v_i^{j'}$ can be a vehicle either in C_i or not in C_i. But $v_i^{j'}$ must be an available fog server both for v_i^j and v_i. $v_i^{j'}$ will execute f_i^j at a very low lazy speed $s_i^{j'}$.

It is important to point out that a helper vehicle helps only one host vehicle at a time. A follower follows only one followee. If one vehicle receives more than one request from the nearby vehicles to help them at the same time, it can only accept one and ignores others.

We need to deal with four situations to ensure reliability: 1) helper computation failure; 2) follower computation failure; 3) helper leave; and 4) follower leave. For the host computation failure, the solution is the same as that of the helper computation failure.

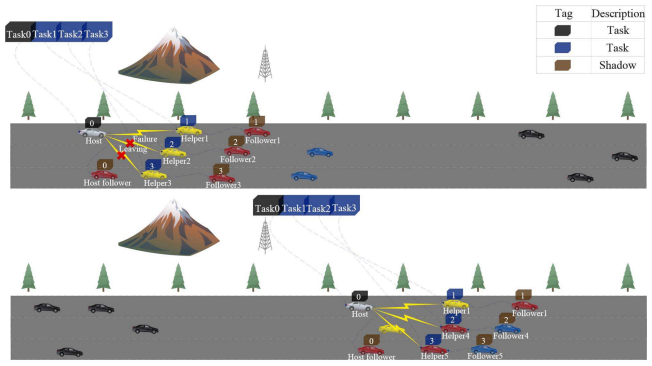

Fig. 3. One example of VVFCA-based network work scenario. There are two different scenarios at two different times. In the time interval, helper2 has one failure and helper3 leaves the cluster.

Helper Computation Failure: Assuming a failure occurs at time t_f on v_i^j, if it has a follower $v_i^{j'}$, and $v_i^{j'}$ is in the available V2V communication range of v_i $v_i^{j'}$ will speed up its execution for f_i^j to its highest speed immediately. At the same time, $v_i^{j'}$ changes to be the main process of f_i^j. It will choose one follower to execute its shadowing replication. This whole process can ensure fast recovery from failure. If v_i^j has no follower when the failure occurs, the task halts.

Follower Computation Failure: If there occurs a failure on a follower $v_i^{j'}$, we just remove it and choose another follower for v_i^j, as long as there are available candidate vehicles.

Helper Leave: As already mentioned, every helper may leave the cluster. Although the host vehicle cannot predict the leave of any helper, it can be informed by the leaver ahead of time. We assume that one helper v_i^j will leave at t_l. It will inform its host and follower $v_i^{j'}$ in advance. If its follower $v_i^{j'}$ is not in the available communication range of v_i, the task halts. Because whether to choose a new helper or return the current task f_i^j to v_i may cause the entire computing of the requirement to time out. If its follower $v_i^{j'}$ is in the available communication range of v_i, $v_i^{j'}$ will work as the new helper of v_i. The follower will speed up its execution for f_i^j to its highest speed immediately. At the same time, v_i^j continues its computing until t_l. $v_i^{j'}$ will choose one follower to execute its shadowing replication. After v_i^j leaves, $v_i^{j'}$ will work as the main processor of f_i^j.

Follower Leave: If a follower $v_i^{j'}$ leaves, we just choose a new follower for v_i^j.

As long as a failure or a helper leave occurs, the corresponding process will be executed. After all task segments have been completed, requirement $\mathscr{R}_i$ is finished. If the total number of failures is larger than a preset threshold $n_i^{\max}$, the requirement will be stated as fail and all processes will be terminated.

We summarize the procedure of our novel PRRM in Algorithm 1. The host vehicle needs to specify $\mathscr{V}_i$ for the set of helpers, $\mathscr{F}_i$ for the set of task segments, and $n_i^{\max}$ for the maximum total number of failures. The execution starts by simultaneously launching $2n$ processes (lines 1 and 2). Each helper has one associated follower. The host also has one associated follower. During the execution, the task management module runs a healthy monitor. The monitor can be implemented by some conventional algorithms such as the heartbeat algorithm. When a failure occurs, the recovery procedure will be executed (lines 6–13). The follower of this fail helper will speed up and take its place as a new helper. The new helper will choose a new follower for itself and launch the follower's process. On the other hand, if one of the helpers wants to leave the cluster (lines 17–23), it will inform the host vehicle and its follower. Its follower will speed up and replaces this helper after it leaves. The new helper will choose a new follower for itself and launch the follower's process. If the total number of failures is larger than $n_i^{\max}$, the whole execution will be terminated (line 5). This process continues until all task segments in $\mathscr{F}_i$ are successfully completed.

Fig. 3 shows one example of VVFCA-based network work scenario. There are 13 vehicles on the highway at the beginning. Four vehicles form a virtual cluster to work for a complex requirement of the host vehicle. The entire task is divided into four task segments which are assigned to the four cars in the cluster. After some time, a failure occurs on helper 2. Its follower (follower 2) will act as the new helper

Algorithm 1 PRRM

Require: $\mathscr{V}_i, \mathscr{F}_i, n_i^{max}$
Ensure: Execution status
1: start $(n-1)$ pairs of helper and follower
2: start one pair of host and follower
3: num = 0
4: Status = "Execution Fails"
5: **while** execution not done **or** $num \leq n_i^{max}$ **do**
6: **if** one failure of helper v_i^j is detected AND it has follower **then**
7: speed up $v_i^{j'}$ and replace the j_{th} helper by $v_i^{j'}$
8: choose one follower for $v_i^{j'}$
9: copy the status of $v_i^{j'}$ to its follower
10: $num = num + 1$
11: **else**
12: Return Status
13: **end if**
14: **if** one failure of follower $v_i^{j'}$ is detected OR one leave message from follower $v_i^{j'}$is detected **then**
15: Select a new follower for v_i^j
16: **end if**
17: **if** one leave message from helper v_i^j is detected AND it has follower in the available communication range **then**
18: speed up $v_i^{j'}$ and replace the j_{th} helper by $v_i^{j'}$
19: choose one follower for $v_i^{j'}$
20: copy the status of $v_i^{j'}$ to its follower
21: **else**
22: Return Status
23: **end if**
24: **end while**
25: **if** $num \leq n_i^{max}$ **then**
26: Status = "Execution Success"
27: **end if**
28: Return Status

(helper 4) of task 2. Helper 4 will choose a vehicle as its follower (follower4). After helper3 leaves the cluster, follower 3 acts as the main process (helper 5) of task 3. Helper 5 will choose a vehicle as its follower (follower 5).

In this section, we innovatively propose a vehicle hardware virtualization solution and integrate it with the V2V communication. This allows vehicles to use their idle resources to serve nearby vehicles while meeting their own needs. We also design a novel reliable collaborative computing scheme among vehicles based on a proactive replication mechanism. This scheme can simultaneously guarantee the computing efficiency and energy efficiency.

IV. Theoretical Analysis

There are a number of challenges and design decisions that need to be addressed when we introduce PRRM into the VVFCA-based network. In this section, two special parameters of PRRM are given: 1) the upper bound on the tolerant total number of failure and 2) difference operators for the progress difference between any two failures. Based on these two parameters, the expected completion time of one complex requirement is quantified. Considering the practical application of PRRM, we give the lower bound on the lazy calculating speed of each follower using the discrete Newton method. All the analyses below are under the assumption that all task segments of requirement $\mathscr{R}_i$ start at time 0, and all task segments of $\mathscr{R}_i$ can be asynchronously executed.

A. Expected Completion Time

Since the helper who is going to leave the cluster will notify its follower and host in advance, its follower can catch up with the progress in time. Therefore, the leave of any helper will not result in any computational delay. The leave of helpers will not affect the whole completion time of complex requirements. However, each failure of helpers will inevitably lead to a delay in the completion time of the whole task. This delay will continue to accumulate. In fact, every requirement in the system should have its maximum tolerant latency which is the so-called time sensitive. We use $\mathscr{T}^i$ to denote the maximum tolerant latency of requirement $\mathscr{R}_i$ and use ${}^rT_i$ to denote the real completion time of $\mathscr{R}_i$. When ${}^rT_i > \mathscr{T}^i$, all results of task segments which are belonging to $\mathscr{R}_i$ must be dropped due to expiration. We assume that T_i represents the completion time of $\mathscr{R}_i$ when there is no failure and helper leaving.

Property 1: For any complex requirement $\mathscr{R}_i$, the completion time of any task segment of $\mathscr{R}_i$ only depends on the vehicle in which it is executed.

Proof: We assume that ${}^rT_i^j$ represents the real completion time of task f_i^j, and ${}^r\mathscr{T}_i = ({}^rT_i^0, {}^rT_i^1, \ldots, {}^rT_i^{n-1})$ represents the set of all task segments' completion time. Because any two task segments are asynchronously executed, we can get that $\forall f_i^j \in \mathscr{F}_i$, and $\forall f_i^k \in \mathscr{F}_i$, ${}^rT_i^j$ is independent to ${}^rT_i^k$. ■

Let $\mathscr{Q}_i = (q_i^0, q_i^1, \ldots, q_i^{n-1})$ denote the set of real delay time of $\mathscr{F}_i$. q_i^j denotes the total delay time of f_i^j, which results from all failures and helper leaving. T_i^j represents the completion time of task f_i^j without any failure and helper leaving, and $\mathscr{T}_i = (T_i^0, T_i^1, \ldots, T_i^{n-1})$ represents the set of all task segments' completion time without any failure and helper leaving. From Property 1, we can get Property 2.

Property 2: The result of $\mathscr{R}_i$ is available if and only if $\max\{T_i^0 + q_i^0, T_i^1 + q_i^1, \ldots, T_i^{n-1} + q_i^{n-1}\} < T_i + \mathscr{T}^i$.

Proof: From Property 1, we know that $\forall f_i^j \in \mathscr{F}_i$ and $\forall f_i^k \in \mathscr{F}_i$, their real completion time is independent. Therefore $\forall f_i^j \in \mathscr{F}_i$, if its real completion time is larger than $T_i + \mathscr{T}^i$, $\mathscr{R}_i$ will time-out. ■

From Property 1, we know that the movement and task execution of vehicles are independent and identically distributed. So the lifetime of n task segments can be modeled as n independent concurrent renewal processes. We assume that $E(X_i^j)$ represents the expected time interval between any two failures of f_i^j, and $N_i^j(t)$ represents the number of failures of f_i^j within time t. From the elementary renewal theorem, we know that

$$N_i^j(t) = \frac{t}{E\left(X_i^j\right)}. \tag{1}$$

We assume that all followers have the same lazy calculating speed. $\forall f_i^j \in \mathscr{F}_i$, $s_i^{j\prime} = s_i'$. From Property 2 and formula (1), we can get the following theorem.

Theorem 1: The system has one *upper bound on the tolerant total number of failure* for each task segment. We denote this upper bound as $n_i^{\max}$. It satisfies

$$n_i^{\max} = \frac{\left(T_i + \mathscr{T}^i\right) - \max_{j\in[0,n]}\left\{T_i^j\right\}}{E\left(X_i^j\right)}. \tag{2}$$

Proof: $\exists f_i^j \in \mathscr{F}_i$ such that $T_i^j = \max_{l\in[0,n]}\{T_i^l\}$. From Property 2, we know that f_i^j has the smallest tolerant expected total number of failures. We denote this number as $n_{ij}^{\max}$. Obviously, $n_i^{\max} = n_{ij}^{\max}$ because of Property 2. The time for these failures is $(T_i + \mathscr{T}^i) - T_i^j$. From formula (1), we can get formula (2). ■

In Algorithm 1, we can input $n_i^{\max}$ as the result of formula (2). We use ${}^lr_i^j$ to denote the recovery time of task f_i^j from its lth failure. We define the difference operators for the progress difference between the lth failure and the $(l+1)$th failure as the following.

Definition 3: The *difference operator* for the progress difference between the lth failure and the $(l+1)$th failure of task f_i^j is denoted by ${}^l\Delta_i^j$. It satisfies

$${}^lr_i^j = \left({}^l\Delta_i^j - \frac{{}^l\Delta_i^j}{{}^ls_i^j}s_i'\right)\frac{1}{{}^{(l+1)}s_i^j}. \tag{3}$$

Here, ${}^ls_i^j$ represents the processing rate of the lth helper of task f_i^j. Fig. 4 shows an example of ${}^l\Delta_i^j$. The computing delay by the lth helper can be calculated by formula (4). $([{}^l\Delta_i^j]/[{}^ls_i^j])$ denotes the calculation time requested for Δ_i^j after the lth failure occurs. $[({}^l\Delta_i^j)/({}^1s_i^j)]$ denotes the calculation time requested for Δ_i^j if the lth failure did not occur

$${}^ld_i^j = \frac{{}^l\Delta_i^j}{{}^ls_i^j} - \frac{{}^l\Delta_i^j}{{}^1s_i^j}. \tag{4}$$

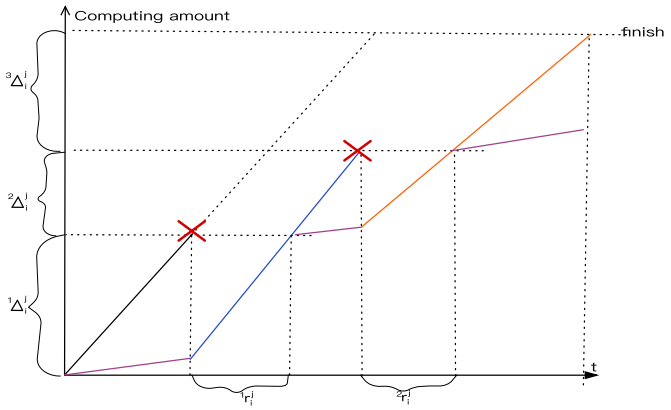

Fig. 4. Example of ${}^{l}\Delta_i^j$ of task f_i^j. There are two failures. Solid black line represents the first helper, solid blue line represents the second helper, and solid orange line represents the third helper. Solid purple line represents the followers with very small rate.

We use p_i^j to denote the probability that f_i^j has a failure in a unit time. Then, $1-p_i^j$ denotes the probability that f_i^j is processed without failure in a unit time. ${}^{l}W_i^j$ represents that there are exactly l failures in the whole computing process of task f_i^j. If the whole computing process is split into ${}^{r}T_i^j$ independent units, ${}^{l}W_i^j$ can be calculated by formula (5) based on the binomial distribution

$$ {}^{l}W_i^j = C_{{}^{r}T_i^j}^{l} \cdot \left(p_i^j\right)^{l} \cdot \left(1-p_i^j\right)^{{}^{r}T_i^j - l}. \tag{5} $$

We use ${}^{e}T_i^j$ to denote the expected completion time of task f_i^j. From formulas (3)–(5), we can get formula (6). The proof of formula (6) will be given in the Appendix. In formula (6), ${}^{e}T_i^j$ consists of three parts. $T_i^j \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j$ is the processing time consumed by helper. $\sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k}([{}^{l}\Delta_i^j]/[{}^{(l+1)}s_i^j] + ((1/[{}^{(l+1)}s_i^j]) - [1/({}^{1}s_i^j)])^{(l+1)}\Delta_i^j)$ is the total processing time consumed by followers. $s_i' \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k}([{}^{l}\Delta_i^j]/[({}^{l}s_i^{j(l+1)}s_i^j)])$ is the lazy processing time consumed by all followers which has been repeatedly counted

$$ \begin{aligned} {}^{e}T_i^j = {} & T_i^j \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j - s_i' \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \frac{{}^{l}\Delta_i^j}{\left({}^{l}s_i^{j(l+1)}s_i^j\right)} \\ & + \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \left(\frac{{}^{l}\Delta_i^j}{{}^{(l+1)}s_i^j} + \left(\frac{1}{{}^{(l+1)}s_i^j} - \frac{1}{{}^{1}s_i^j} \right)^{(l+1)} \Delta_i^j \right). \end{aligned} \tag{6} $$

For the simplicity of discussion, we ignore the time which is used to consolidate the results of all split tasks and the time which is used to transfer messages between vehicles. From Property 1, we can get the following theorem.

Theorem 2: If ${}^{e}T_i$ represents the expected completion time of $\mathscr{R}_i$

$$ {}^{e}T_i = \max_{j\in[1,n]} \left({}^{e}T_i^j \right). \tag{7} $$

Proof: From Property 1, we know that the values of ${}^{e}T_i^j$ and ${}^{e}T_i^k$ are independent with each other for any two task segments f_i^j and f_i^k. Because all task segments are executed asynchronously, the maximum execution time of all task segments is the execution time of the whole requirement. ■

B. Lower Bound on the Lazy Calculating Speed of Followers

As already mentioned, followers can execute at a very low speed to save energy. However, from formulas (6) and (7), we know that the initial execution speed of the follower s_i' is inversely proportional to the expected completion time of the whole requirement. Obviously, the lower s_i' is, the more energy the corresponding follower can save in the whole process, and the more delay will be introduced, and *vice versa*. In application, s_i' can be derived by balancing the tradeoffs between delay and energy. In this section, we present the lower bound on s_i' from a series of mathematical analysis of formula (7).

Because ${}^{r}T_i^j \geq T_i^j$, the system will have a greater lower bound of s_i' when we change ${}^{r}T_i^j$ to T_i^j in formula (5). Moreover, we can get T_i^j at the beginning. Therefore, we use T_i^j to calculate the approximate value of ${}^{l}W_i^j$ by

$$ {}^{l}W_i^j \approx C_{T_i^j}^{l} \cdot \left(p_i^j\right)^{l} \cdot \left(1-p_i^j\right)^{T_i^j - l}. \tag{8} $$

We use $\Delta_i^j = \{{}^{1}\Delta_i^j, {}^{2}\Delta_i^j, \ldots, {}^{n_i^{\max}}\Delta_i^j\}$ to denote the set of all different operators of f_i^j. For the simplicity of discussion, we define A_i^j, $f(\Delta_i^j)$, and $g(\Delta_i^j)$ as follows:

$$ A_i^j = T_i^j \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \tag{9} $$

$$ f\left(\Delta_i^j\right) = \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \frac{{}^{l}\Delta_i^j}{\left({}^{l}s_i^{j(l+1)}s_i^j\right)} \tag{10} $$

$$ \begin{aligned} g\left(\Delta_i^j\right) = {} & \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \left(\frac{{}^{l}\Delta_i^j}{{}^{(l+1)}s_i^j} \right. \\ & \left. + \left(\frac{1}{{}^{(l+1)}s_i^j} - \frac{1}{{}^{1}s_i^j} \right)^{(l+1)} \Delta_i^j \right). \end{aligned} \tag{11} $$

By substituting formulas (9)–(11) into formula (6), we can rewrite formula (6) as ${}^{e}T_i^j = A_i^j + g(\Delta_i^j) - s_i' f(\Delta_i^j)$. From Theorem 2, we can get

$$ {}^{e}T_i = \max_{j\in[1,n]} \left(A_i^j + g\left(\Delta_i^j\right) - s_i' f\left(\Delta_i^j\right) \right). \tag{12} $$

Here, A_i^j is a constant. We consider ${}^{e}T_i$, $g(\Delta_i^j)$, and $f(\Delta_i^j)$ as functions of s_i'. ${}^{e}T_i$ is rewritten as ${}^{f}T_i(s_i')$. Then, we can rewrite formula (12) as a function of s_i'

$$ {}^{f}T_i\left(s_i'\right) = \max_{j\in[1,n]} \left(g\left(\Delta_i^j\right) - s_i' f\left(\Delta_i^j\right) + A_i^j \right). \tag{13} $$

Obviously, for any fixed ${}^{l}\Delta_i^j$ and ${}^{l}s_i^j$ ($l \in [0,n]$), $g(\Delta_i^j) - s_i' f(\Delta_i^j) + A_i^j$ is a linear decreasing function on s_i'. Hence, we have Theorem 3 is true. Its proof will be given in the Appendix.

Theorem 3: Function ${}^{f}T_i(s_i') = \max_{j\in[1,n]}(g(\Delta_i^j) - s_i' f(\Delta_i^j) + A_i^j)$ is a piecewise linear decreasing function with a unique root ${}_{*}s_i'$.

From Theorem (3), we know that ${}^{f}T_i(s_i') - (\mathscr{T}_i + T_i)$ is also decreasing and has only one root ${}_{*}s_i'$. It is the lower bound of the calculating speed of followers.

Theorem 4: For requirement $\mathscr{R}_i$, ${}_{*}s_i'$ is the *lower bound of the calculating speed of followers*.

Proof: $\mathscr{T}_i + T_i$ is the maximum tolerant completion time of $\mathscr{R}_i$ and is constant. Function ${}^{f}T_i(s_i')$ is piecewise linear decreasing. Thus, ${}^{f}T_i(s_i') - (\mathscr{T}_i + T_i)$ is also piecewise linear decreasing. Then $\forall s_i' > {}_{*}s_i', {}^{f}T_i(s_i') - (\mathscr{T}_i + T_i) < 0$. Therefore $\forall s_i' > {}_{*}s_i', {}^{f}T_i(s_i') < (\mathscr{T}_i + T_i)$. This means that when $\forall s_i' < {}_{*}s_i'$ the requirement $\mathscr{R}_i$ can never be completed within the deadline. ∎

As already mentioned, the discrete Newton method can get the root of piecewise linear decreasing function with the number of iteration is polynomial in the input size. Based on the discrete Newton method, we design the lower bound searching algorithm for followers' calculating speed which is shown as Algorithm 2.

The set of difference operators Δ_i^j cannot be got before execution. We can use history values as its approximation (line 1). For each iteration, we use the greedy algorithm to find ${}_{u}\Delta$ with which the expected completion time has the maximum value (line 5). If the root is not reached, then ${}_{u}s_i'$ can be got by the Newton Method in each iteration (line 11).

Algorithm 2 Lower Bound Searching Algorithm

Input: $\mathcal{T}_i, T_i, (T_i^1, T_i^2, \ldots, T_i^n), s_i^j$
Output: Lower bound of s_i'

1: Get the history Δ_i^j
2: $A = \max(T_i^1, T_i^2, \ldots, T_i^n)$
3: $u = 1, {}_{u}s_i' = 0, state = False$
4: While $state = False$
5: Find ${}_{u}\Delta$ of ${}^{f}T_i(s_i')$ by greedy algorithm
6: $b_u = {}^{f}T_i(s_i')$
7: **if** $b_u = 0$ **then**
8: $\quad state = True$
9: **else**
10: $\quad u = u + 1$
11: $\quad {}_{u}s_i' = \frac{s_i g({}_{u}\Delta) + A_i^j}{f({}_{u}\Delta)}$
12: **end if**
13: end While
14: Return ${}_{u}s_i'$

V. Performance Evaluation

In this section, we will show the simulations of proposed algorithms. The experiment preparation will be discussed with parameter settings in Section V-A. Results analysis will be given in Section V-C.

A. Experimental Setup

To validate the performance of PRRM, we do some performance evaluations. In our simulation, we consider the highway VANET environment. We randomly generate traffic on the highway. Ten vehicles (hosts) with one complex computation requirement and 100 vehicles (helpers and followers) with random amounts of free computing resources are randomly scattered on the highway. We assume that the largest V2V communication distance is R. One complex requirement will be randomly split into l segments. l vehicles that have free resources and within the available communication distance of one host vehicle are randomly chosen as helpers for this host. One helper will choose one follower from the vehicles that have free resources and within the available communication distance. Here, having available communication distance means that the distance between the two communication sides is less than R. When failures or leaves occur, new helpers and followers will be selected according to the same rules as above. All processes are simulated by MATLAB. In the following discussion, the number of task segments equals to l. All task segments will be assigned among vehicles optimally: vehicles with more computing resources get task segments with larger sizes. The two typical events (failure and vehicles leave) in the system follow the Poisson distribution. All results are obtained from the average results of 1000 repetitive. All results are normalized.

To show the performance of our proposed PRRM algorithm, we compare it with the central cloud-based (CC) algorithm [29] and the double-head clustering (DHC) algorithm [30].

CC Algorithm: This algorithm is a general centralized algorithm. Vehicles send their computation requirements and initial data to the central cloud. The cloud is responsible for complex calculation and returns results to vehicles.

DHC Algorithm: This algorithm is a full backup algorithm. To make the vehicle collaboration cluster more stable, the algorithm chooses two functioning cluster heads which backup for each other in a single cluster.

B. Evaluation Metrics

Various evaluation metrics are used in our experiments.

Reliability: The main purpose of collaborative computing in the VANETs is to help vehicles quickly complete complex requirements with limited resources in a limited time. We dispatch some complex requirements to the host vehicles which are beyond their capability. If the system is not reliable enough, collaborators cannot guarantee to submit its collaborative computing results to the host vehicles in time. Much time will be wasted on finding new collaborators and coordinating task assignments after computation failures or vehicle leaves occur. Therefore, we use the completion rate of complex requirements in a limited time to measure the reliability of the proposed algorithm.

Efficiency: We use the total power consumption (TPC) and the total execution time (TET) of complex requirements to measure the energy efficiency of our proposed algorithm.

Because the energy consumption in this system is mainly electricity power consumption, we use TPC to represent the energy consumption in the experiments. Here, TPC $= E_{ho} + E_{he} + E_f$. E_{ho} is the TPC of the host. E_{ho} is the TPC of helpers. E_f is the TPC of followers.

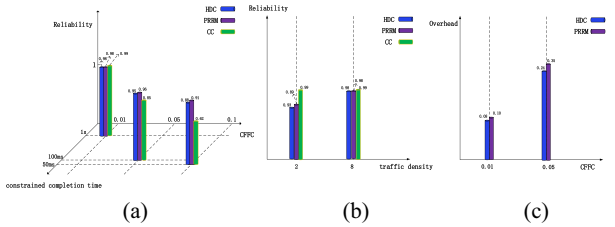

Fig. 5. Reliability and overhead related results.

C. Results Analysis

Reliability: In collaborative VANET, computation failure probability, frequency of vehicles' route changing, and traffic density are the three main factors that affect the system reliability. We use CFFC to represent the computation failure probability and the frequency of vehicles' route changing of each vehicle. The effects of CFFC and traffic density on reliability will be discussed in the following.

Fig. 5(a) shows the reliability comparison results among PRRM, CC, and DHC under different CFFC and constrained completion time. In these experiments, the traffic density is set to a high level. When the CFFC is very low (0.01) and the constrained completion time is long (1 s), the reliability of CC is the best. PRRM is slightly weaker than DHC. Because the communication time is sufficient, and there are few failures and leaves. The reliability of the central cloud outperforms that of the mobile fog servers. However, when the CFFC is larger (0.05) and the constrained completion time is shorter (100 ms), some client vehicles no longer have enough time to exchange data with the cloud server. The reliability of CC significantly decreases. The gain of DHC and PRRM emerges. Because closer collaborative computation is more time saving than the remote central cloud. When the CFFC is 0.1 and the constrained completion time is 10 ms, PRRM gets the top performance. Computation failures will cause many computation restarts. This will lead to many computation time out in DHC and CC.

Fig. 5(b) shows the effect of traffic density. We set constrained completion time as 100 ms and CFFC as 0.01. For PRRM, we simply use the number of helpers associates with each host vehicle, which is described as l above to describe the traffic density. For DHC and CC, the corresponding traffic densities are $(1/2) + 1$ and $(1/2)$ separately. When the traffic density is large, all algorithms perform well. When the traffic density is low, DHC and PRRM have poor performance. For host vehicles, there are few vehicles in their available V2V communication ranges in this situation. Then, they can hardly find enough helpers. They complete their computing tasks mainly by their own computing resources. Therefore, collaborative VANET algorithms are more suitable for high-density traffic scenarios.

As a brief summary of reliability comparison, PRRM is more reliable when the mobility of vehicles is higher and traffic density is lower.

Energy Efficiency: We continue to increase the number of task segments from 1 to 10 to observe the TPC and TET. Results are compared with that of DHC.

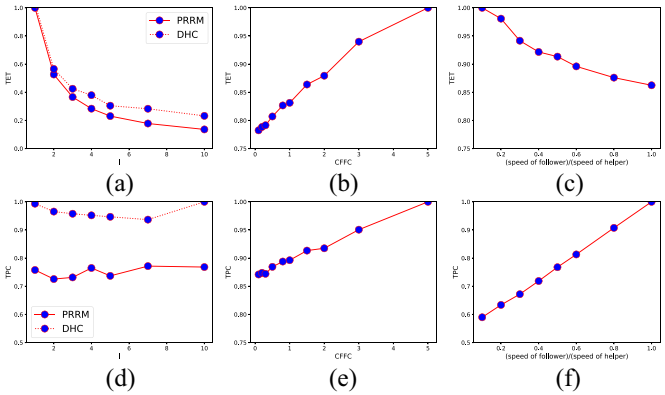

Fig. 6. Energy efficiency results.

Fig. 6(a) and (b) shows the experimental results. We set the probability of failure as 0.05. The ratio of the calculating speed of followers to that of helpers is set to 0.6. Because we ignore the energy consumption of message communication and the failure probability is very small, the energy consumption of two mechanisms has only minor ascension as the number of task segments increases. At the same time, the number of cluster header exchanging in DHC is very small. Thus, its TPC is smaller than our mechanism. This means DHC consumes less energy than our mechanism for the same requirement when the failure probability is very small. But when some vehicles in the cluster leave, especially the cluster header leaves, many task segments must be reexecutated from the beginning under DHC. The energy consumption and time consumption are unacceptable. But for our mechanism, the extra energy consumption is very small. Therefore, when the failure probability is high, our mechanism is energy efficient while ensuring reliability.

As the number of task segments increases, the average TET of the two different mechanisms drops rapidly. It is reasonable. Because more task segments will activate more vehicles to calculate for the same requirements collaboratively. A computing restart after a failure always takes a lot of time. This time is much larger than the time it takes for the follower to catch up on the progress. Therefore, the TET of our mechanism is smaller. As a brief summary of energy efficiency metrics comparison between PRRM and DHC, PRRM has higher energy efficiency under all scenarios.

Relationship Between the Probability of Failure and TPC: In Fig. 6(c) and (d), we show the change in the average TET and TPC of the system as the probability of failure increases from 0.1% to 5%. The ratio of the calculating speed of followers to that of helpers is set to 0.6. The increasing of the probability of failure means that there are more computing failures or helper leaving from the cluster during the whole collaborative computing of complex requirements. The calculation accelerating process of followers will be activated more times to ensure that task segments can be completed correctly. This process is time consuming, so the rise in the TET curve is reasonable. The TPC will increase accordingly.

Effect of the Calculating Speed of Followers: We continue to increase the ratio of the calculating speed of followers to

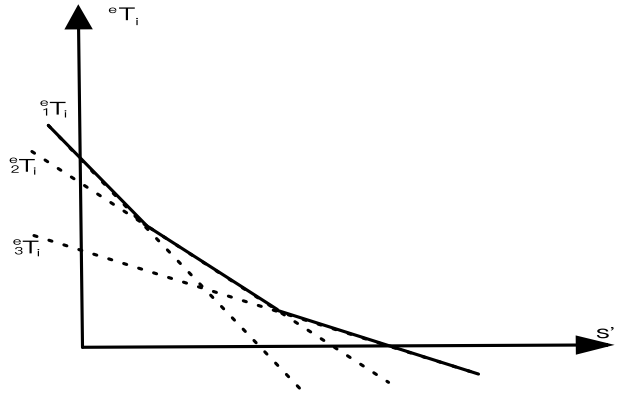

Fig. 7. Example of $^{f}T_i(s'_i)$. Solid line is the segments which have the maximum values of $g(\Delta_i^j) - s'_i f(\Delta_i^j) + A_i^j$.

that of helpers from 0.1 to 1.0. Fig. 6(e) and (f) shows the change of the average TET and TPC under our mechanism. The number of task segments is 12. The increasing of the ratio means the decrease of the time that is used by followers to catch up the progress of the helpers which have failures. We can see from the figure that as the followers' calculation accelerates, the average TET is decreasing. At the same time, TPC has the opposite trend. It increases rapidly as the calculation speed of followers increases. A higher calculation speed consumes more energy. So it is very meaningful to calculate the lower bound of the calculating speed of followers.

Overhead: The overhead mainly comes from Hello packets messages, cluster creation messages, cluster maintenance messages, communication messages, and synchronization messages. We use the average total amount of all messages but the core data exchange in the execution process of a requirement as the overhead of compared algorithms. Fig. 5(c) shows the overhead comparison results between PRRM and DHC. The first observation of this figure tells that the higher CFFC, the higher overhead in PRRM and DHC. This is reasonable. A higher CFFC refers to a higher probability of vehicle leaves. Then, more clusters need to recoordinate their structures. There will be more cluster creation messages and cluster maintenance messages in PRRM and DHC. The overhead of PRRM is a little higher than DHC in the situation with high CFFC. Because more role-exchange messages between helpers and followers are needed to maintain the cluster. These messages are nonexistent in DHC.

VI. Conclusion

In this article, we address the reliability problem in collaborative VANET. First, we define VVFCA for VANET. By virtualizing vehicles as fog servers and enabling nearby vehicles to communicate with V2V links, this new architecture brings many advantages. Second, we design PRRM to improve the reliability of the system. Through theoretical analysis, we get the upper bound on the tolerant total number of failures. The expected completion time and the lower bound on followers' calculating speed are also present. The simulation results demonstrate that our proposed algorithm can improve the reliability and save energy, compared to other algorithms. In future work, we will study situations that one helper vehicle helps more than one host vehicle at the same time, and that one follower follows more than one followee at the same time.

Appendix A
Proof for Formula (6)

We assume that the maximum number of failure of task f_i^j is $n_i^{\max}$. p_i^j denotes the probability that f_i^j has a failure in a unit time. The probability that there is only one failure is

$$^{1}W_i^j = C^1_{^{r}T_i^j} \cdot p_i^j \cdot \left(1 - p_i^j\right)^{^{r}T_i^j - 1}.$$

The probability of that there are exactly two failures is

$$^{2}W_i^j = C^2_{^{r}T_i^j} \cdot \left(p_i^j\right)^2 \cdot \left(1 - p_i^j\right)^{^{r}T_i^j - 2}.$$

The probability of that there are exactly three failures is

$$^{3}W_i^j = C^3_{^{r}T_i^j} \cdot \left(p_i^j\right)^3 \cdot \left(1 - p_i^j\right)^{^{r}T_i^j - 3} \ldots$$

The probability of that there are exactly $n_i^{\max}$ failures is

$$^{n_i^{\max}}W_i^j = C^{n_i^{\max}}_{^{r}T_i^j} \cdot \left(p_i^j\right)^{n_i^{\max}} \cdot \left(1 - p_i^j\right)^{^{r}T_i^j - n_i^{\max}}.$$

The probability that there is no failure occurs is

$$^{0}W_i^j = 1 - \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j.$$

The recovery time from the lth failure is

$$^{l}r_i^j = \left(^{l}\Delta_i^j - \frac{^{l}\Delta_i^j}{^{l}s_i^j} s'_i\right) \frac{1}{^{(l+1)}s_i^j}.$$

The computing delay by the lth new helper is

$$^{l}d_i^j = \frac{^{l}\Delta_i^j}{^{l}s_i^j} - \frac{^{l}\Delta_i^j}{^{1}s_i^j}.$$

Therefore, the expected completion time of f_i^j is

$$\begin{aligned}
&{}^{0}W_i^j \cdot T_i^j + {}^{1}W_i^j \cdot \left(T_i^j + {}^{1}r_i^j + {}^{1}d_i^j\right) \\
&\quad + \cdots + {}^{n_i^{\max}}W_i^j \cdot \left(T_i^j + \sum_{l=1}^{n_i^{\max}} \left({}^{l}r_i^j + {}^{l}d_i^j\right)\right) \\
&= T_i^j \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j + \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \left({}^{l}r_i^j + {}^{l}d_i^j\right) \\
&= T_i^j \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j - s'_i \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \frac{^{l}\Delta_i^j}{\left(^{l}s_i^j {}^{(l+1)}s_i^j\right)} \\
&\quad + \sum_{k=1}^{n_i^{\max}} {}^{k}W_i^j \sum_{l=1}^{k} \left(\frac{^{l}\Delta_i^j}{^{(l+1)}s_i^j} + \left(\frac{1}{^{(l+1)}s_i^j} - \frac{1}{^{l}s_i^j}\right) {}^{(l+1)}\Delta_i^j\right).
\end{aligned}$$

APPENDIX B
PROOF OF THEOREM 3

For any fixed ${}^{l}\Delta_i^j$ and ${}^{l}s_i^j$ ($l \in [0, n]$), because $f(\Delta_i^j) > 0$, $g(\Delta_i^j) - s_i' f(\Delta_i^j) + A_i^j$ is a linear decreasing function on s_i' (dash lines in Fig. 7).

For any interval of s_i', there must be a set of ${}^{l}\Delta_i^j$ in Δ_i^j which makes $g(\Delta_i^j) - s_i' f(\Delta_i^j) + A_i^j$ has the maximum value. These line segments consist of ${}^{f}T_i(s_i')$.

Fig. 7 shows an example of ${}^{f}T_i(s_i')$. The dash line is some $g(\Delta_i^j) - s_i' f(\Delta_i^j) + A_i^j$ when we fix ${}^{l}\Delta_i^j$. The solid line represents ${}^{f}T_i(s_i')$.

REFERENCES

[1] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the IoT environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1587–1595, May 2014.

[2] J. Pereira, L. Ricardo, M. Luís, C. Senna, and S. Sargento, "Assessing the reliability of fog computing for smart mobility applications in VANETs," *Future Gener. Comput. Syst.*, vol. 94, pp. 317–332, May 2019.

[3] Y. Qu *et al.*, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.

[4] E. Borcoci, M. Vochin, and S. Obreja, "Mobile edge computing versus fog computing in Internet of Vehicles," in *Proc. 10th Int. Conf. Adv. Future Internet*, 2018, pp. 8–15.

[5] 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.

[6] X. Cheng, L. Yang, and X. Shen, "D2D for intelligent transportation systems: A feasibility study," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1784–1793, Aug. 2015.

[7] T. Thirugnanam and M. R. Ghalib, "A reward based connectivity-aware IoV neighbor selection for improving reliability in healthcare information exchange," in *Peer-to-Peer Networking and Applications*. Cham, Switzerland: Springer, 2020, pp. 1–11.

[8] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, and D. Z. Du, "Reliability-aware offloading and allocation in multilevel edge computing system," *IEEE Trans. Rel.*, early access, May 15, 2019, doi: 10.1109/TR.2019.2909279.

[9] D. Zhang, H. Ge, T. Zhang, Y.-Y. Cui, X. Liu, and G. Mao, "New multi-hop clustering algorithm for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1517–1530, Apr. 2019.

[10] I. Ahmad *et al.*, "VANET–LTE based heterogeneous vehicular clustering for driving assistance and route planning applications," *Comput. Netw.*, vol. 145, pp. 128–140, Nov. 2018.

[11] A. A. Khan, M. Abolhasan, and W. Ni, "An evolutionary game theoretic approach for stable and optimized clustering in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4501–4513, May 2018.

[12] Y. Zhang, H. Zhang, K. Long, Q. Zheng, and X. Xie, "Software-defined and fog-computing-based next generation vehicular networks," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 34–41, Sep. 2018.

[13] 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.

[14] R. Vilalta *et al.*, "Control and management of a connected car using SDN/NFV, fog computing and YANG data models," in *Proc. 4th IEEE Conf. Netw. Softw. Workshops (NetSoft)*, 2018, pp. 378–383.

[15] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *Proc. IEEE Integr. Netw. Manag. (IM) Int. Symp. IFIP/IEEE*, 2015, pp. 1202–1207.

[16] A. Soua and S. Tohme, "Multi-level SDN with vehicles as fog computing infrastructures: A new integrated architecture for 5G-VANETs," in *Proc. IEEE 21st Conf. Innov. Clouds Internet Netw. Workshops (ICIN)*, 2018, pp. 1–8.

[17] J. C. Nobre *et al.*, "Vehicular software-defined networking and fog computing: Integration and design principles," *Ad Hoc Netw.*, vol. 82, pp. 172–181, Jan. 2019.

[18] S. Park and Y. Yoo, "Network intelligence based on network state information for connected vehicles utilizing fog computing," *Mobile Inf. Syst.*, vol. 2017, Jan. 2017, Art. no. 7479267.

[19] C. Tang, X. Wei, C. Zhu, W. Chen, and J. J. Rodrigues, "Towards smart parking based on fog computing," *IEEE Access*, vol. 6, pp. 70172–70185, 2018.

[20] Y. Xiao, Z. Ren, H. Zhang, C. Chen, and C. Shi, "A novel task allocation for maximizing reliability considering fault-tolerant in VANET real time systems," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2017, pp. 1–7.

[21] H. Min, W. Seo, J. Lee, S. Park, and D. Hong, "Reliability improvement using receive mode selection in the device-to-device uplink period underlaying cellular networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 2, pp. 413–418, Feb. 2011.

[22] D. Feng, L. Lu, Y. Yuan-Wu, G. Y. Li, G. Feng, and S. Li, "Device-to-device communications underlaying cellular networks," *IEEE Trans. Commun.*, vol. 61, no. 8, pp. 3541–3551, Aug. 2013.

[23] Y. Meng, Y. Dong, C. Wu, and X. Liu, "A low-cost resource re-allocation scheme for increasing the number of guaranteed services in resource-limited vehicular networks," *Sensors*, vol. 18, no. 11, p. 3846, 2018.

[24] H. Chour, Y. Nasser, H. Artail, A. Kachouh, and A. Al-Dubai, "VANET aided D2D discovery: Delay analysis and performance," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8059–8071, Sep. 2017.

[25] T. Radzik, "Parametric flows, weighted means of cuts, and fractional combinatorial optimization," in *Complexity in Numerical Optimization*. Singapore: World Sci., 1993, pp. 351–386.

[26] F. A. Radu and I. S. Pop, "Newton method for reactive solute transport with equilibrium sorption in porous media," *J. Comput. Appl. Math.*, vol. 234, no. 7, pp. 2118–2127, 2010.

[27] E. Riks, "The application of Newton's method to the problem of elastic stability," *J. Appl. Mech.*, vol. 39, no. 4, pp. 1060–1065, 1972.

[28] T. Makela and S. Luukkainen, "Incentives to apply green cloud computing," *J. Theor. Appl. Electron. Commerce Res.*, vol. 8, no. 3, pp. 74–86, 2013.

[29] M. A. Salahuddin, A. Al-Fuqaha, M. Guizani, and S. Cherkaoui, "RSU cloud and its resource management in support of enhanced vehicular applications," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2014, pp. 127–132.

[30] G. H. Alsuhli, A. Khattab, and Y. A. Fahmy, "Double-head clustering for resilient VANETs," *Wireless Commun. Mobile Comput.*, vol. 2019, Feb. 2019, Art. no. 2917238.

Luobing Dong received the Doctoral degree from Xidian University, Xi'an, China, in 2013.

He was a Visiting Scholar with the University of Texas at Dallas, Richardson, TX, USA. He is an Associate Professor with the School of Computer Science and Technology, Xidian University. His research interests include big data processing, mobile cloud computing, and document summarization.

Qiufen Ni is currently pursuing the Ph.D. degree with the School of Computer, Wuhan University, Wuhan, China.

She is also a joint Ph.D. student with the University of Texas at Dallas, Richardson, TX, USA, from October 2019 to October 2020. Her research interests include wireless networks, VANET, social networks, theoretical approximation algorithm design, and blockchain.

Weili Wu (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, in 2002 and 1998, respectively.

She is currently a Full Professor with the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA. Her current research interests include data communication, data management, the design and analysis of algorithms for optimization problems that occur in wireless networking environments, and various database systems.

Chuanhe Huang (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 1985, 1988, and 2002, respectively.

He is currently a Professor and the Ph.D. supervisor with the School of Computer, Wuhan University. His research interests include SDN, opportunistic networks, and wireless networks, focusing on cryptography, wireless security, and trust management.

Taieb Znati (Member, IEEE) received the M.S. (Computer Science) degree from Purdue University, West Lafayette, IN, USA, in December 1981, and the Ph.D. (Computer Science) from Michigan State University, East Lansing, MI, USA, in 1988.

He joined the Faculty of the Department of Computer Science with a joint appointment in the Graduate Program of Telecommunications, University of Pittsburgh, Pittsburgh, PA, USA, in 1988. His research interests include distributed multimedia systems, high-speed networks to support real-time applications, performance evaluation, and local area networks.

Ding Zhu Du (Member, IEEE) received the Doctoral degree from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 1985.

He is a Professor with the Department of CS, University of Texas at Dallas, Richardson, TX, USA. His research interests include design and analysis of approximation algorithms for combinatorial optimization problems with applications in computational biology and computer and communication networks.