An infrastructure-assisted job scheduling and task coordination in volunteer computing-based VANET

Abdul Waheed¹ · Munam Ali Shah¹ · Abid Khan² · Gwanggil Jeon^{3,4}

Received: 30 October 2020 / Accepted: 9 June 2021
© The Author(s) 2021

Abstract

Vehicular networks as the key enablers in Intelligent Transportation Systems (ITS) and the Internet of Things (IoT) are key components of smart sustainable cities. Vehicles as a significant component of smart cities have emerging in-vehicle applications that can assist in good governance for sustainable smart cities. Most of these applications are delay sensitive and demand high computational capabilities that are provided by emerging technologies. Utilizing the distributed computational resources of vehicles with the help of volunteer computing is an efficient method to fulfill the high computational requirements of vehicles itself and the other components of smart cities. Vehicle as a resource is an emerging concept that must be considered to address the future challenges of sustainable smart cities. In this paper, an infrastructure-assisted job scheduling and task coordination mechanism in volunteer computing-based VANET called RSU-based VCBV is proposed, which enhances the architecture of VANET to utilize the surplus resources of vehicles for task execution. We propose job scheduling and task coordination algorithms for different volunteer models. Further, we design and implement an adaptive task replication method to seek fault tolerance by avoiding task failures due to locations of vehicles. We propose a task replication algorithm called location-based task replication algorithm. Extensive simulations validate the performance of our proposed volunteer models while comparing average task execution time and weight ratios with existing work.

Keywords Intelligent transportation systems · Smart cities · Vehicles as a resource · Job scheduling · Task replication

Introduction

Sustainable and diverse services such as traffic management, health care, education, public safety, etc. are being provided by smart cities. For successful and sustainable smart cities,

enhancement and efficiency while providing services are two significant ingredients. Traditional cloud and edge solutions are experiencing more latencies and additionally a negative impact on acquiring smart green city solutions. With the immense increase of the Internet of Things (IoT) devices, traditional solutions are increasing the load on the infrastructures and servers dedicated to multiple offloading services. The increased load and the delay induced due to the increased access to the same access point from multiple users also affect the latency [1]. Smart cities are becoming more popular due to easy access to advanced technologies. These cities are simply using connectivity in different departments [2]. To fulfill the user requirements, services provided by smart cities are depending on emerging technologies such that IoT, communication between devices, fog computing, etc. A smart city is not merely an infrastructure but the level at which these infrastructures assist in attaining sustainable development targets [3].

Vehicular network as one key enabler in Intelligent Transportation Systems (ITS) has earned great interest in the provision of road safety, and comfort. It can play a significant

✉ Gwanggil Jeon
ggjeon@gmail.com
Abdul Waheed
gallian92@gmail.com
Munam Ali Shah
mshah@comsats.edu.pk
Abid Khan
abk15@aber.ac.uk

¹ Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan

² Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, Wales, UK

³ School of Electronic Engineering, Xidian University, Xi'an 710071, China

⁴ Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, South Korea

role in the provision of safety and comfort in smart cities with the help of emerging applications and new developments in technologies [4]. In recent years, the number of vehicles is also growing day by day increasing the requirements to access the remote servers for emerging applications in Vehicular Ad hoc Networks (VANETs). These networks support user safety, comfort, and more emerging applications like autonomous driving, augmented reality, etc. These applications are inducing an upsurge of vehicles' computational demands. With the help of IoT and related emerging technologies, improvement can be made in environmentally sustainable developments [5]. To satisfy these increasing demands, cloud computing is conflated with vehicular networks for the new domain Vehicular Cloud Computing (VCC). This domain mainly focused on utilizing vehicle resources with the help of static and random vehicular clouds.

Mobile Cloud Computing (MCC) used remote cloud servers for task offloading from mobile devices. Moving data to distant clouds for task executions causes higher latency which was not adequate in delay-sensitive applications. The reasons behind the confrontation of drawbacks are the far distance of the cloud servers and the overload of the backhaul networks [6]. Since data are being increasingly produced at the user level at the edge of the network which makes a high transmission latency for cloud-based processing of data. Therefore, processing data at the edge of the network would be more efficient. Edge computing refers to distributed computing bringing computation performance at the edge of the network [7]. Shifting of cloud server functionalities to the edge of the network for mobile devices is a new trend for delay-sensitive applications. Edge computing plays a key role in improving the latency for delay-sensitive applications needs for good governance in sustainable smart cities [8].

In vehicular networks, VCC and MCC are centralized services provided from distant clouds with high transmission latency. Mobile edge computing (MEC) is a new paradigm for mobile devices with computation at proximities. MEC has been come forth as a key technology to facilitate the computation workloads of mobile computing devices. It also lowers the makespan incurred due to computation-intensive applications. It does not complement but supplements the cloud computing functionality by using edge servers at base stations to execute the computational tasks offloaded from mobile devices having a cellular connection. In this work, jobs from Smart Mobile Devices (SMDs) are offloaded to MEC server or locally executed on SMD's [9]. Vehicular

edge computing (VEC) is a new promising domain where computations are shifted to the edge of the network. In VEC, servers are deployed in proximity near vehicles to benefit the roadside unit (RSU) which is usually in the communication range of vehicles [10]. VEC has the same concept of supplementation of cloud using edge server but not base stations but at RSUs. VEC is dedicated mostly for vehicles and vehicles have DSRC to connect to the RSUs. In VEC, jobs from vehicles are offloaded to RSUs for execution. It started from placing a single server in proximity between cloud servers and vehicles. Due to load on a single server, balancing the load on multiple servers was the new step. Due to the static server, there was a shorter link duration between vehicles and servers. This was addressed using mobile MEC servers placed on vehicles moving in traffic [10].

The fog computing concept was put forward by Cisco before edge computing [11]. Their main purpose was the same, bringing cloud computing capabilities to the edge of the network for delay-sensitive applications. The fog computing paradigm enables computing at the edge of the network where data are originally being generated. Moreover, it can be used as a framework for reducing the delays in IoT devices used in sustainable smart cities [12]. These are not replacing cloud but the extension of cloud computing [13]. Many research [14, 15] used these terms alternatively despite the small differences like task responsibility, focus, handling multiple IoT, etc., [16]. Vehicular fog computing (VFC) is a new paradigm that came into existence by the integration of fog computing and vehicular networks [17]. Solutions were presented by shifting the task execution in proximity like edge or fog computing [18]. Increased task execution requirement calls for the additional server deployment to address the load which incurs significant capital and operational expenditures. With the advancement in technology, vehicles are being equipped with advance digital systems which are giving birth to a new trend of using vehicles as resources to fulfill the advanced needs of task execution. A vehicle can upload or download huge files using the resources of nearby vehicles. Driving analysis and surrounding hurdles can be detected with the help of nearby vehicles in less time in comparison to the offloading request to remote servers.

Volunteer Computing-Based VANET (VCBV) is a new emerging concept for utilizing surplus resources in VANET using volunteer computing [19]. Due to the high computational requirements of modern applications, a single vehicle cannot fulfill the task execution requirements. In VCBV, task execution is performed using the surplus resources

of multiple vehicles simultaneously. There are different modes to apply VCBV for task execution. In our previous work, we have presented the idea of resource utilization in VANETs focusing on taxonomy, applications, scenarios, and challenges whereas in this article we extend our work by proposing algorithms for one of its types, i.e., RSU-based task coordination. An infrastructure-based method will be applied to accomplish job scheduling and task coordination in VCBV. Since infrastructures used in VANETs are Road-Side Units (RSUs), therefore, these will be used to adopt the entire procedure. Accomplishing multiple jobs with different priorities requires some central entity. Therefore, RSU-based VCBV is used where jobs are sorted according to their priorities to improve weight ratio which ad hoc coordination lacks. Applying VCBV requires handling many challenges to achieve the up to mark performance. First, the deadline for the jobs and the user status of the job initiator affect the job scheduling process. Many factors are required to be considered while selecting the job during VCBV process. Second, the task partitioning and coordination process comprise accessing the resources and capabilities of candidate volunteers. Third, due to the dynamic nature of the vehicular networks, it is more challenging to use the resources in an optimal way when the network topology is changing every fraction of a second. Therefore, fault tolerance and efficient schemes are required to replicate the tasks anticipated to be failed due to certain reasons. Given these challenges, it is required to design job scheduling and task coordination algorithms for RSU as the job scheduler and task coordinator. There is a need to design an efficient task replication algorithm to avoid the fault toleration incurred due to task failure. The following are the contribution of the article.

- We propose a model deployed on an RSU that utilizes the surplus resources of vehicles in RSU's range. This model accomplishes two achievements, i.e., it utilizes the surplus resources of vehicles and it executes the low latency jobs without relying on third-party paid services.
- To meet the deadline requirements of delay-sensitive low latency jobs, we propose job scheduling and task coordination algorithm which schedules the jobs according to nature of the job and the requesting entity and after acquiring the information and willingness of volunteers, it coordinates the partitioned tasks to eligible volunteers. Simulation results demonstrate that proposed algorithm achieves a 33% improvement in average execution time for 30 tasks, an average of 12% performance improvement for all number of tasks from 10 to 30 tasks and

30% improvement in the weight ratio for 50 tasks when compared with a multi-objective algorithm [20].

- A fault-tolerant task replication method is proposed. It consists of an algorithm which helps to avoid the delay incurred due to task failures. It replicates the tasks to avoid task failures. Furthermore, simulation results demonstrate that the proposed LTR technique has shown a 43% better performance than the existing multi-objective algorithm [20] for exactly 30 tasks. For all number of tasks from 10 to 30, an average of 21% performance improvement is achieved.

The remainder of this paper is structured as follows: “[Related works](#)” summarizes the related works. “[Volunteer computing-based VANET](#)” briefs the concept of volunteer computing-based VANET and its types. Architecture and system model for RSU-based VCBV are presented in “[RSU-based VCBV](#)”, followed by the overview of the solution and proposed RSU-based job scheduling, task coordination and task replication algorithms are discussed in “[Proposed RSU-based job scheduling and task coordination](#)”. Performance evaluation is done in “[Performance evaluation](#)”, and finally, we conclude in “[Conclusions and future work](#)”.

Related works

In this section, we describe the literature about task execution in vehicles, infrastructure participation, and task replication effects on scheduling process. With the advancement in technology, the number of connected devices is increased tremendously. This increment has more possibilities for corporate projects like volunteer computing. However, these corporate efforts are seriously affected by some security concerns like privacy. Panadero et al. [21] developed “the multicriteria biased randomized method (MBRM)” to pick out nodes assuring a minimal quality of service for the users. This method is founded on a lexicographic ordering multicriteria strategy. The key idea of this technique was distributing and balancing the load for all the nodes. An actual social network was used to simulate the results to select the reliable nodes. In cloud computing, services such that SaaS, PaaS, and IaaS are offered according to the pay-as-you-go model. VCC and MCC both use the services of clouds in different manners and both can be leveraged in vehicular networks.

MCC is a computing paradigm where processing is offloaded to servers outside the computing devices. Some basic advantages of MCC include the extension in battery

life, better processing, more reliability, dynamic provisioning, and scalability. Wu et al. [22], proposed an Energy-Efficient Algorithm (EEA) based on Lyapunov optimization which optimizes the energy efficiency by switching the offloading between local, cloud, and cloudlet computing. Due to the facility of executing tasks in proximity, edge computing is used to reduce latency in vehicles using the offloading mechanism. Computations offloading is a process where all or some parts of an application are migrated to other processing entities [23]. However, the basic functionality of computation offloading is the decision about the task to be offloaded or not, and the server where it would be offloaded [24]. Offloading applications to the cloud depend upon two factors those are cloud connectivity and availability. Offloading is affected by low resources of bandwidth and more network access latency.

MCC has addressed the challenges of constrained resources in mobile computing devices with high latency and network congestion tradeoffs due to the distant clouds. To address these issues, MEC is put forward with the placement of a host server at the edge near to mobile computing devices. Wang et al. [25], proposed “EdgeFlow”, a processing system for utilizing the computation services from all over the network. The proposed system was a multilayer data flow model for the minimization of transmission delay while using all the network entities, i.e., cloud, MEC server, edges devices, etc. Partial offloading is an effective decision in MEC to achieve low latency in real-time applications. To reduce the time delay in MEC, an iterative heuristic algorithm was designed for resource allocation [26]. In another work [27], multiple edge users using non-orthogonal multiple access in MEC to offload the edge server and other cellular users. While minimizing the average execution time, both communication and computation costs are considered.

MCC and VCC are promising computing paradigms for the vehicles due to the high processing power of clouds. However, the network bandwidth with a great load of additional communication induces the issue of high latency. This issue was raised due to remote clouds and load on network bandwidth which is addressed using fog computing where computing is shifted closer to end-users [28]. These services are hosted in the proximity of end devices fulfilling the demand for low latency. As the architecture of the cloud and edge is growing, its energy requirements are increasing resulting in carbon pollution. Therefore, a part of the research in VEC computation offloading focused on energy

efficiency as a framework named BEGIN, which is the integration of big data analysis and VEC. The demonstration of the feasibility of the proposed framework is presented with the help of a case study [29]. A framework named Chimera [30] was proposed for energy reduction in vehicles having crowdsensing applications. Deciding the optimal portion of the task to be offloaded based on states of energy for efficiency was presented for in-vehicle user equipment. A multi-objective VEC task scheduling algorithm is proposed for task offloading from user vehicle to MEC vehicle. Extensive simulations show less task execution time for task offloading with high reliability [20]. Tasks are offloaded to mobile MEC servers placed on vehicles moving in traffic. These servers are additionally mounted on special vehicles moving in traffic. Vehicles offload jobs for execution to these servers. These servers work fine for a smaller number of jobs but upon increasing number of tasks, these servers start dropping the jobs which increases the failure ratio affecting the average execution time. Also, the existing offloading services include the monetary costs which the requesting vehicles have to pay even the free resources in nearby vehicles are wasting.

Fog layer is placed between the cloud and the end-users where delay-sensitive tasks can be offloaded instead of offloading them to far clouds. However, another task with more computational requirements can be sent to the cloud for execution at the same time. Fog computing and vehicular network combinedly can use the surplus resources of vehicles in the form of vehicular fog nodes. In VFC, computational task offloading can be performed with moving or parked vehicular fog nodes. Hou et al. [31], presented the concept of VFC where vehicles are utilized as infrastructure. This new domain is based on the collaborative utilization of communication and computation resources of several edge devices or end-user devices. Due to the wide geographical distribution of fog computing, VFC is a better option for delay-sensitive applications in vehicular networks [32]. Ibrahim et al. [33] used the emerging concept of Vehicle-as-Resource (VaaR) to utilize the surplus resources in fog-enabled vehicular networks. This work addressed the issue of job priorities while acquiring lower makespan with higher throughputs of jobs accomplishments. An algorithm named Dantzig–Wolfe is used to solve the problems and validated using numerical analysis. However, in this work, the resources are utilized by offloading a job to a single vehicle which limits the selection of jobs for execution to those jobs that can be accomplished by a single OBU.

Table 1 Comparison of features of existing schemes

Features	MBRM [21]	MCC [22]	VEC [29]	MEC [26]	VFC [31]	RSU-based model (pro- posed)
Is service free of cost (\$)?	✓	✗	✗	✗	✗	✓
Low latency	✗	✗	✓	✓	✓	✓
Applicable without internet connectivity	✗	✗	✓	✗	✓	✓
Local resource utilization	✗	✓	✗	✗	✓	✓
Without additional cost for deployment	✓	✗	✗	✗	✗	✓
Local monitoring	✗	✗	✗	✗	✓	✓
Job partitioning	✗	✗	✗	✗	✗	✓
High network bandwidth	✗	✗	✓	✓	✓	✓
Delay-sensitive applications	✗	✗	✓	✓	✓	✓
Geographically distributed	✓	✗	✗	✗	✓	✓
Real-time application support	✗	✗	✓	✓	✓	✓
User proximity	✗	✗	✓	✓	✓	✓

Task execution failure is the major cause of increased makespan in task scheduling. Usually, a task failure is mitigated by retrying or using alternative resources for the same task which is task replication [34]. When task replication is applied, it uses the additional resources for the successful completion of the task, but it sometimes fails. In parallel computing when dependent tasks are offloaded to different hosts, the delayed or failed results from some nodes can affect the whole process of job completion. Replicating tasks on different nodes is one way to combat the job completion delay due to task failures or delays [35]. VCC utilizes the surplus resources of vehicles for sharing the load of the cloud data center. To meet the deadline and avoid task failure, the tasks are replicated to multiple vehicles in vehicular clouds. The policy uses the least number of replicas to meet the deadline of computational tasks in encounter-based VCC [36]. Since the system is centralized, frequent signaling is required for updation which is an overhead. In another work [37], learning-based task replication is done to minimize the offloading delay in vehicular edge computing. The internal failure in the cloud data center can delay the completion of a big data application like scientific workflows. To mitigate the internal or external failure of the cloud, a fault-tolerant algorithm was proposed by Li et al. [38]. Experiments prove the effectiveness of the proposed task replication-based algorithm.

Table 1 shows the comparison of features of the proposed RSU-based model with the works of related domains.

Volunteer computing-based VANET

The new approach emerged from the integration of volunteer computing and VANET is named VCBV. The need for the new emerging approach raised when there was a requirement

to fulfill the computational needs with the help of surplus resources in VANET. In VCBV, vehicles with surplus computational and storage resources offer services voluntary. When a job is requested by a vehicle or any other entity from outside VANET, it is served by the volunteer vehicles within the VANET collectively. Job initiator requests for the job execution by offloading jobs to other entity which can be a vehicle or an RSU. This is the job coordinator which is responsible for the job partitioning into tasks and then task coordination. Generally, VCBV is categorized into two types depending upon the nature of the job coordinator. In ad hoc VCBV, vehicles are job initiators and also responsible for the job coordination process. In RSU-based VCBV, infrastructures like RSUs are the central entities for volunteer computing acting as job coordinator. These two types are explained as follows.

Ad hoc VCBV

In VANET, sometimes, vehicles might not be connected to the Internet which makes it inappropriate for task offloading to remote servers. However, the presence of OBUs makes the vehicles eligible for Vehicle-to-Vehicle (V2V) communication using DSRC. Using V2V communication, vehicles can offload their tasks to nearby vehicles. This type of VCBV where no infrastructure like RSU is not used is called ad hoc VCBV. Low latency is the biggest outcome for this type of VCBV. This happens because of the provision of computing solutions in proximity. Vehicles moving in the same direction is the best use case of ad hoc VCBV. There are more chances of resource utilization in highway scenarios. This type of VCBV is also applicable to vehicles in congestion. Even, if vehicles are parked or stopped, their computing powers can be utilized without the connectivity to the Internet.

In this type of computing, the job initiator and the job coordinator are the same that is a vehicle with tasks to be executed. The first hurdle in starting the process is the eligibility of the vehicle for becoming the job initiator. The problem is who will decide the status of the job initiator. The second issue is getting a willingness from the nearby vehicles for being the volunteers. Finally, the coordination of partitioned tasks to eligible volunteers requires an efficient algorithm for the successful completion of jobs. In this article, we will address these issues by proposing an appropriate algorithm and validating through experiments. In this article, we are covering RSU-based VCBV while leaving ad hoc VCBV for future work.

RSU-based VCBV

VCBV can be used in volunteer with or without the use of RSU. In ad hoc task coordination, the selected job initiator starts the VCBV process without the discrimination of priority of jobs. Accomplishing multiple jobs with different priorities requires some central entity. Therefore, RSU-based VCBV is used where jobs are sorted according to their priorities to improve weight ratio which was not possible in ad hoc coordination. In this type of computing, RSU behaves like a central entity responsible for all job scheduling and task coordination using VCBV. RSU-based computing avoids the problem of job initiator selection and enhances the procedure by adding a job scheduling which is more

Table 2 List of notations

Notation	Description
T_r	Wireless transmission rate between RSU and vehicles
tp_i	A tuple for task representation
tp_{id}	Task identity
tp_{is}	Task input size
tp_{icr}	Computational resources required for task tp_i
C_i	Volunteer vehicle's computational capability
T_i	Makespan (Time taken for a task to complete)
I_i	The number of incentives a vehicle i has
r	Any positive value assigned for incentivization such as 1
O_1	First objective function
O_2	Second objective function
LET	Link expiration time
V_i	Vehicle i
w_i	Total weight attained after completion of all the jobs

effectual for delay-sensitive jobs. In this type, vehicles are only volunteers ready for the execution of tasks assigned by RSUs. Job executed in RSU-based VCBV can be in archive of RSU previously offloaded or freshly assigned from vehicles. Multiple RSUs can be involved depending upon the size and nature of the tasks coordinated. The architecture of RSU-based VCBV is shown in Fig. 1,

Fig. 1 RSU-based VCBV Architecture

Fig. 2 System model

RSU-based VCBV

In this section, we present our proposed RSU-based VCBV architecture and elaborate on the system model in detail. The important notations used in this paper are summed up in Table 2.

System model

The scenarios considered in this paper are of moving or parked vehicles lying in the range of some nearby RSU. The model of RSU-based VCBV is described in Fig. 2. Details are as follows.

Job initiator

In RSU-based VCBV, the job initiator can be any vehicle from the parking lot or moving on roads, a passenger from public transport, or any IoT device. A vehicle that has some job to be performed is a job initiator. Any vehicle, pedestrian, a worker from a nearby office that needs resources for some job to be executed is a job initiator.

Job coordinator

In RSU-based VCBV, the job coordinator is RSU which a central identity and a static node in VANET. When an RSU receives the jobs from job initiators, it schedules the job according to priority and job initiator's gained status. The

selected job then partitioned into executable tasks and coordinated to eligible volunteer vehicles.

Volunteer vehicle

These are the vehicles with surplus resources to perform task execution and are agreed for participation in volunteer computing in conjunction with other volunteers. These vehicles are usually in the communication range of RSU at the time of task coordination. RSU is responsible for the adjustment of the number of volunteers for voluntary depending upon the size and nature of tasks to be performed. We assume that RSU has n number of vehicles in its communication range those are agreed to being volunteers. Volunteer vehicles in communication range and willing to perform volunteer computations are denoted as $N = \{1, 2, 3, \dots, n\}$.

Communication model

In the presented scenario, the communication is between the job initiator and RSU then between RSU and volunteer vehicles. As per the IEEE 802.11p standard, 3Mbps to 27Mbps data rates are supported over 10 MHz bandwidth [39]. For wireless communication, carrier-sense multiple access with collision avoidance which involves the Request-To-Send (RTS) and Clear-To-Send (CTS) for collision reduction. The whole VCBV network works on the radio spectrum that is allocated for communication between RSU and vehicles. The transmission rate between the RSU and volunteer vehicles can be determined by applying Shannon's formula [40] which is as follows.

$$T_r = \delta \log_2 (1 + \text{SNR}), \quad (1)$$

where T_r is the transmission rate achieved between RSU and vehicles, δ is the portion of bandwidth allocated to each link and SNR is the signal-to-noise ratio and can be explained as follows.

$$\text{SNR} = \frac{P \times d}{N \times \delta + I_r}, \quad (2)$$

where P is transmission power, d is the distance between RSU and vehicle, N is the density of the power spectral, and I_r is interference due to noise. It must be considered that SNR depends upon the change in distance between the RSU and volunteer vehicles [41].

Job scheduling

Job priority is one of the significant matters during the scheduling process because some of the jobs must be performed earlier to meet a deadline. It must be considered as an important factor in suitable job scheduling algorithms. There were two phases of a vehicle in ad hoc VCBV, i.e., a job coordinator and a volunteer. In RSU-based VCBV, the involvement of RSU diminishes the complications generated due to the two phases of a vehicle. When RSU is involved VCBV has two tasks, i.e., job scheduling and task coordination. A job is a set of many independent or dependent tasks. If we denote a job with “ J ”, then the possible set of n jobs can be shown as follows.

$$\text{Job}(n) = \{J_1, J_2, J_3, \dots, J_i \dots J_n\},$$

where n is any natural number.

A single job “ J_i ” can be described as the following tuple.

$$J_i = \{J_{ID}, J_{UT}, J_{PT}, J_{AT}\},$$

where

J_{ID} is ID of job J_i

J_{UT} is the type of user who initiated the job. Users are differentiated by the number of incentives (I_i) they have.

J_{PT} is Job priority type depending on the urgency and deadline of the job.

J_{AT} is the job arrival time ($J_{AT} = I$).

Here, we can calculate the priority P_i for a job i using the following formula

$$P_i = \alpha \times J_{UTi} + \beta \times J_{PTi} - \gamma \times J_{ATi}, \quad (3)$$

where P_i is the priority of job J_i and α, β , and γ are weights, where $\alpha + \beta + \gamma = 1$.

Task model

After the job scheduling process, RSU gets the willingness from volunteer vehicles and partitions the first job into executable tasks according to the number of volunteers ready for voluntary. Each task can be represented as tuple $tp_i = \{tp_{i_{ID}}, tp_{i_{IS}}, tp_{i_{CR}}\}$, where “ i ” is the vehicle number from 1 to n willing for voluntary, $tp_{i_{ID}}$ is the identity of the task sent, $tp_{i_{IS}}$ is the input size of that task and $tp_{i_{CR}}$ denotes the required computational resources for computing task t_i .

Computation model

The computation-intensive task is divided into three major time components such that task transmission time, computation time, and results collection time. Task transmission time depends on the size of the task, i.e., $tp_{i_{IS}}$, bigger the task size the time taken to transmit to volunteers is larger and vice versa. The computation time of the task depends on two factors, namely as $tp_{i_{CR}}$ and the computational capability of the volunteer vehicle, i.e., C_i . The time taken for transmission of output from volunteer to job initiator depends on the size of output data and this is usually very small and negligible. The total time taken for a task from transmission time to completion of task and results collection is shown in the following equation.

$$T_i = \frac{tp_{i_{IS}}}{T_r} + \frac{tp_{i_{CR}}}{C_i}. \quad (4)$$

Task replication

In VCBV task coordination, a delay in performance can happen due to internal failure (host volunteer failure). Even a single task failure can cause a reasonable delay in job completion. A significant delay affects the target completion time of the job which reduces the service reliability of the system. Fault tolerance and performance can be achieved by exploiting the redundancy of surplus computing resources using task replication. In this method, task replicas are offloaded to multiple volunteer vehicles while completing an important job. A task replication method generally increases resource utilization and can cause a higher delay if not applied smartly. Knowing the tasks that are required to be replicated is the key to designing a smart task replication algorithm for fault tolerance and performance. In RSU-based VCBV, most of the task failures are happen due to the loss of connectivity to the volunteers that can happen due to out of range. Analyzing the volunteers with higher probabilities of task failures due to out-of-range movement is the basic aim of the task replication algorithm.

Incentivization

In VCBV, vehicles share their surplus resources for the completion of tasks assigned from neighboring vehicles. Since the nature of volunteer computing is to rely on the resources of different vehicles, resources are always unreliable regarding willingness and job completion issues. There is a need to motivate the vehicles to keep volunteering until completion of tasks. Incentivization is a process in which users are rewarded for their contributions. In VCBV, incentives are given to the volunteer vehicles as points after the completion of a job which can later be used when that vehicle is going to offload some jobs for executions. If “ I_i ” is the number of incentives a vehicle “ i ” has then after the completion of one successful task, it will be as follows.

$$I_i = I_i + r, \quad (5)$$

where “ r ” can have any positive value such as one.

Proposed RSU-based job scheduling and task coordination

In this section, we shall concentrate on an overview of the solution for job scheduling and task offloading to volunteer vehicles using RSU-based VCBV.

Overview of the solution

As per the communication and computational models explained earlier, the offloading decisions for a job coordinator are coupled. When several jobs have been received by the job coordinator, there is a need to schedule the jobs according to priorities, job initiator status, and the deadline for job completion. The availability of resources and balanced resource allocation are other factors that need to be kept considered because unbalanced resource allocation strategies lead to low performance and job failures. Completing a job after its deadline is also considered as job failure.

In this article, we consider two optimization goals: weight-age of the completed jobs according to their priorities plus deadlines and the average execution time for completed jobs. Let T_i be the total time to complete the job “ i ” and “ w ” be the total weight for all completed jobs. The job scheduling and task coordination scheme will be optimized if it minimizes the total task execution time and maximizes the weight ratio. Our problem is based on the achievement of these two goals while pinpointing the possible constraints. Therefore, the objectives and constraints for job scheduling and task coordination in RSU-based VCBV can be formulated as the following.

$$O_1 : \min \sum_{i=1}^n T_i, \quad (6)$$

$$O_2 : \min \sum_{i=1}^n w_i, \quad (7)$$

Fig. 3 Three-lane highway RSU-based scenario

where T_i is makespan and w_i total weight attained after completion of all the jobs. If any coordination algorithm fulfills the aforesaid objectives while handling the subsequent constraints, it will be considered as a suitable algorithm. The constraints that need to be satisfied by the job scheduling and task coordination algorithms are computation and communication constraints. A vehicle cannot perform any computation requiring more resources than it acquires and the link expiration time (LET) between two job initiator must be more than the time taken to complete the task execution by the volunteer as illustrated in the following equations.

$$C1 : tp_{iCR} < C_i, \quad (8)$$

$$C2 : LET < \frac{tp_{iCR}}{C_i}. \quad (9)$$

RSU-based job scheduling and task coordination

We consider a three-lane highway where vehicles are moving in the same direction as shown in Fig. 3.

A job initiator can be a vehicle, a pedestrian, or an IoT device with a single or more jobs that need to be performed whereas volunteers are the vehicles in the communication range of RSU. These volunteer vehicles are having surplus resources and are ready to execute the tasks. In RSU-based VCBV, after the arrival of more than one job, RSU calculates the priorities for submitted jobs and schedule these jobs for execution. As explained before, the priorities calculated depend on many factors affecting the order of jobs to be performed. This algorithm lacks the scheduling of jobs using the priority of jobs which will be handled in future work.

The job coordinator (RSU) uses periodical beacon frames to identify volunteer vehicles in proximity. To avoid communication channel overload, only one hop vehicles are considered voluntary. When a job initiator (passenger or vehicle) needs the CPU-based computations, and this job cannot be

executed on a single OBU or takes more than the deadline time. RSU receives all the requests where these jobs are queued regarding their priorities. These jobs are partitioned one by one into executable tasks and then these tasks are coordinated to eligible volunteers by RSU. RSU-based scenario is well organized compared to ad hoc VCBV due to the involvement of the central entity which increases the completion of urgent and important tasks. To complete the whole offloading process of RSU-based VCBV, we propose a job scheduling and task coordination algorithm named JSTC. Like ad hoc task coordination, arrivals, and departures of the volunteer vehicles in volunteer groups can happen and affect the performance. The arrival of the vehicles does not affect the RSU-based JSTC process until the completion of tasks for a specific job. However, the departure of a vehicle due to any of the reasons before the completion of a running job can affect the performance of VCBV. This can be handled by reassigning the failed tasks to other vehicles in the next round which compromises the efficiency and increases the average job execution time. The other method of handling the failure tasks is the task replication strategy which assures fault tolerance.

There are some assumptions for the proposed algorithms. The assumptions are as follows:

- If a vehicle is receiving the beacon frames, it is a volunteer and vehicles are for participation in volunteer computing.
- All vehicles are equipped with the hardware required for the communication and computation process [42].
- Channel conditions and physical layer parameters are ideal [43].
- The job assigned is compatible with OBU for execution and can be partitioned into multiple tasks.
- No malicious vehicle is present in the system and security parameters are set.

Algorithm 1: Proposed JSTC algorithm for RSU based VCBV**Input:** Jobs $J(n)$ **Output:** Scheduled list of jobs, Average Job Execution Time (AJET)**BF**→ beacon frames used by RSU to synchronize with volunteers**BFR**→ beacon frames response used to get acknowledgment from willing volunteers**TC**→ Task Coordinator **J_{pmax}** → Maximum priority for any job **V_n** → Number of Volunteers **R_i** → Roadside unit i **JET**→ Job Execution Time1: **while** $J(n) \neq \text{NULL}$ **do**2: **for** $i = 1 \rightarrow J(n)$ **do**3: Find J_{UTi} 4: Find J_{PTi} 5: Find J_{ATi} 6: Calculate $P_i = \alpha \times J_{UTi} + \beta \times J_{PTi} - \gamma \times J_{ATi}$ 7: **end for**8: Sort P_i in descending order9: $T_1 = \text{time.now}()$;10: **for** $j = J_{pmax} \rightarrow 1$ **do**

11: Select job for coordination

12: Make R_i TC13: **while** $(V_n < n)$ **do**

14: Broadcast BF

15: Get BFR

16: **end while**17: Partition the job J_j into n tasks18: **for** $k = 1 \rightarrow n$ **do**19: Send tp_k to V_k 20: **end for**21: **for** $k = 1 \rightarrow n$ **do**22: Receive RR from V_k 23: Send r to V_k 24: **end for**

25: Arrange RR

26: Aggregate RR

27: $T_2 = \text{time.now}()$ 28: $T_m = T_2 - T_1$ 29: $JET = JET + T_m$ 30: **end for**31: $AJET = JET/m$

32: Return AJET

33: **end while**34: **end**

Table 3 Simulation parameters

Parameter	Value
Simulator	NS-3.27
Number of job coordinators (RSU)	1
Number of volunteer vehicles	5,10,20
Beacon frame size	20 B
Data packet size	1024 B
Result packet size	200 B
Transport Protocol	UDP
OBU computation rate	[0.5, 1] GHz
Data rate	6Mbps
PHY	IEEE 802.11p
Frequency	5.9 MHz
Bandwidth	10 MHz
RSU communication range	300 m
V2V communication range	[125–250] m
Vehicles velocity	[60,80] km/h
Highway length	1000 m

After the successful transmission of tasks to eligible volunteer vehicles, completion of the job relies on the successful completion of the task and submission of RR back to the job coordinator (RSU). In a dynamic environment of VANET, the uncertain movement of vehicles can lead to task failure resulting in a delay in job completion. This delay can lead to violation of the deadline which is a basic feature of job completion. Minimizing the

deadline violation probability requires the minimization of task failures that can be worked out using task replication techniques. It is a technique that allows a single task to be executed on multiple volunteers.

There are two types of modes to apply a task replication technique to RSU-based VCBV systems. In the first mode, every task will be assigned to more than one volunteer exactly, i.e., exactly two replicas for every task. In this case, there low probabilities for task failures and resubmission of tasks but with higher resource usage. In the second mode, tasks are smartly allocated to volunteers by replicating only tasks with chances of failures. Since most of the chances of task failures are due to out of range new locations of volunteer, volunteers can be chosen on behalf of their recent locations for task replication.

For the task replication algorithm, we update the beacon frame contents with the velocity of vehicles and distance from RSU. For simplicity, we assume that vehicles are moving on straight roads and there is no change of direction for some time. Predictions about future locations can be utilized to analyze the probability of failure. Only those volunteers are chosen for replication which are expected for being out of range before the completion of jobs. We propose a Location-based Task Replication (LTR) algorithm to address the delay in job completion due to task failures. We assume that all partitioned tasks have the same execution time for all the volunteer vehicles. The task execution time for all the tasks is the same but the failure of tasks can happen due to the failure of volunteer vehicles.

Fig. 4 Average execution time with a different number of tasks

Algorithm II: Proposed LTR algorithm for RSU based VCBV*Input: Scheduled list of jobs**Output: Average Job Execution Time (AJET)**BF*→ beacon frames used by RSU to synchronize with volunteers*BFR*→ beacon frames response used to get acknowledgment from willing volunteers*TC*→ Task Coordinator*Vi* → Vehicle*Vn*→ Number of Volunteers*JET*→ Job Execution Time*Ri* →Roadside unit *i*1: *T1*=time.now()2: **for** *j*=*Jpmax* → 1 **do**

3: Select job for coordination

4: Make *Ri* TC5: **while** (*Vn*<*n*) **do**6: Broadcast *BF*7: Get *BFR*8: **end while**9: *s*←number of volunteers expected for failure10: Partition the job *Jm* into *n-s* tasks11: **for** *k*=1→*n-s* **do**12: Send *tpk* to *Vk*13: **end for**14: **for** *k*=*n-s+1*→*n* **do**15: Send *tpk* to *Vk*16: **end for**17: **for** *k*=1→*n* **do**18: Receive *RR* from *Vk*19: Send *r* to *Vk*20: **end for**21: Arrange *RR*22: Aggregate *RR*23: *T2*=time.now()24: *Tm*=*T2-T1*25: *JET*=*JET*+*Tm*26: **end for**27: *AJET*=*JET*/*m*28: Return *AJET*

29: End

Performance evaluation

In this section, we conduct simulations to evaluate the proposed RSU-based VCBV. We perform our experimentation on NS3, a discrete event simulator for networks having models for all layers. It is an open-source simulator most widely used in academia for communication models [44]. Our experimental model consists of one RSU and vehicular nodes and Wireless Access in Vehicular Environments

(WAVE) as the system architecture. This architecture is used for the VANET environment with an extension to 802.11, which is 802.11p. Vehicles deployment is uniform in three lanes and their velocities are from 60 to 80 km/h. The parameters of the network simulator are shown in Table 3 [45].

Usually, a job is completed by a vehicle using the Local Computation (LC) or it can be offloaded to other vehicles to minimize the job completion time. In the

infrastructure-assisted environment, RSU is a centralized entity to manage job scheduling. The job initiator is any vehicle, passenger, pedestrian, or any IoT device outside the VANET. RSU is the job coordinator who is responsible for the complete process consisting of initialization, task allocation, and result collection.

Task computational model

A computational task can be handled locally or be offloaded to RSU present in the communication range. The local computational model and volunteer models are as follows.

Local computing

The initiator vehicle uses its own OBU to compute the tasks locally. The results obtained from this algorithm are a benchmark for evaluation and it is a kind of threshold for the task execution time [20]. The average task execution time of any of the protocols approaching or greater than LC indicates the failure of that protocol.

Multi-objective vehicular edge computing task scheduling algorithm

This algorithm is used to optimize the allocation of communication and computing resources jointly for VEC. In this work, multiple servers were installed on vehicles to fulfill the computational need of user vehicles. Multiple

vehicles offload their computational tasks to respected servers and wait for the results. The concept of using surplus resources of user vehicles is very close to existing work where all the executions are done on a single vehicle. We shifted the load of task executions on multiple vehicles instead of a single vehicle. The work presented in [20], is quite similar and validates the makespan for job execution which is the primary contribution for our work where a job is distributed into multiple volunteers to improve the makespan.

Volunteer models

In this type of computing, tasks are offloaded to a different number of vehicles for volunteer computing. More volunteers mean that the job is to complete with more resources. We consider different scenarios using 5, 10, and 20 volunteer vehicles. We represent these as 5 V, 10 V, and 20 V. These volunteer models follow the rules proposed in FTC.

Straight task replication model

Task replication is a technique used to achieve fault tolerance. In the straight task replication technique, each task is executed more than once but on different volunteer vehicles. We use an RSU-based 20 V-2R model where 20 volunteers are used and each task is replicated twice. In other models, the same number of volunteers are used

Fig. 5 Weight ratio for different number of tasks

Fig. 6 Average execution time for varying cluster radius **Fig. 7** Weight ratio for varying cluster radius

Fig. 8 Average execution time for STR with a different number of tasks

Fig. 9 Average execution time for STR with different cluster radius

Fig. 10 Average execution time for LTR with a different number of tasks

Fig. 11 Average execution time for LTR with different cluster radius

with each task having three replications for each task on different volunteer vehicles named RSU-based 20 V-3R.

Location-based task replication

In this model, task replication is used smartly only for those tasks which are lying beyond some specific location and have the probability to move out of range. Aforementioned all the volunteer models are used with LTC. These models are RSU-based 5 V-LTR, RSU-based 10 V-LTR, RSU-based 20 V-LTR.

Job coordinator and volunteer communication model

In this model, the job initiator/coordinator is a vehicle which has to do computational offloading to volunteer vehicles. Initially, the communication comprises four basic types of relaying which are as follows.

Beacon frame

A Beacons Frame (BF) is the request for volunteering from RSU to the volunteer vehicles. It contains information about the capacity and size of the task data. It also contains the number of operations (OpsNum) and the size of the data to be sent (DataSize). The volunteer must know the data size for reliability. Only the job coordinator is broadcasting the BFs, all other volunteers are potential recipients.

Beacon frame response

When the volunteer receives the BF, it checks its availability and capacity. If the volunteer is available and ready for the computations, it sends the Beacon Frame Response (BFR). Replying with BFR means that the vehicle is agreed for voluntary and it has all the required resources to be utilized and it will be available until the job is done. It also sends its location and velocity of the vehicle.

Task assign

When the job coordinator receives the willingness of the volunteer it sends the Task Assign (TA) packet to volunteers. It may be a single packet or multiple packets depending on the size of the task data. After sending all the TA packets to the volunteers, it waits for the reply from the volunteers.

Result reply

When a job is completed by the volunteer node. It sends the Result Reply (RR) to the job coordinator. The RSU when

receives the RR, it checks for the number of tasks completed and retransmits the TA in case of failed tasks. After the completion of all the tasks, it aggregates the result.

Performance comparisons

In this section, we evaluate the performance of LC- and RSU-based volunteer groups in three different scenarios. In these scenarios, we compare the parameters of average execution time and weight ratios. In the first scenario, average execution time and weight ratios for a different number of tasks are compared. In the second scenario, these same parameters are compared for a fixed number of tasks but different cluster radius between the job coordinator and volunteer vehicles without task replication.

Different number of tasks

In this scenario, the radius of the vehicle cluster is fixed to 100 m and the number of tasks is varied from 10 to 50. First, we compare the average execution time for a varied number of tasks.

It is observed from Fig. 4 that the highest delay is incurred by LC which executes all parts of the job locally on a single OBU. The highest average execution time makes LC not beneficial for delay-sensitive jobs. Besides LC, other algorithms use offloading technique to perform the job executions. In RSU-based algorithms, RSU offloads the task to volunteer vehicles whereas multi-objective algorithm offloads to MEC servers. The multi-objective algorithm is better in average execution time when compared with RSU-based 5 V. It is also better than RSU-based 10 V where the tasks are less than 30. When the number of tasks is more than 30, the average execution time for multi-objective is increasing showing more delays. RSU-based 20 V algorithm has the lowest delay regarding job execution time when compared with the rest of the algorithms. The reason for this performance is increased computational capability. Therefore, it is concluded that the higher the number of volunteers, the lower is the average execution time for offloaded jobs. For the RSU-based 20 V model, a decrease of 7%, 25%, and 46% is achieved for 10, 25, and 50 tasks, respectively.

The weight ratio depends on the order of job execution according to their priorities. Higher will be weight ratio if the jobs with higher priorities are accomplished earlier than the jobs with lower priorities. Figure 5 shows that the multi-objective algorithm has the highest weight ratio until the completion of offloading of almost half of the tasks. The reason for this good ratio is the use of multiple servers for task offloading. A higher number of servers induce very good results for fewer tasks but when the number of tasks is increased, multi-objective shows low performance comparing to proposed algorithms. Therefore, it is concluded that

the RSU-based 20 V algorithm shows 10% and 30% better weight ratio performance for 35 and 50 tasks, respectively.

Variable cluster radius

In variable cluster radius, average execution time and weight ratio are determined for cluster radius of 50–300 m; whereas, the number of tasks is 30 for all algorithms. The comparison for these parameters is shown in Figs. 7 and 8. Likewise the first scenario, the LC algorithm has the worst performance. The reason behind the low performance of the LC algorithm is that it does not offload the jobs to any server or volunteer model. This is because the LC algorithm does not offload any part of the jobs to any server of volunteer models. It has no effect of varying cluster radius because there is communication with other vehicles. The increase in cluster radius between the RSU and volunteers always affect the communication due to lower SNR. Due to this decrease, there is an increase in transmission time which affects the performance. In Fig. 6, it can be observed that the average execution time is constant until the radius becomes 125 m between the job coordinator and volunteers. With the increased cluster radius, the average task execution time increases for all the algorithms. When the distance between the RSU and volunteer vehicles is increased, the packet dropping ratio increases which induce the retransmission resulting in higher delay. It is concluded that RSU-based 20 V has the least impact on task execution time with varying cluster radius. RSU-based 20 V has a 33% better performance when compared with a multi-objective algorithm. Figure 7 shows the comparison of the weight ratios for 30 tasks with varied cluster radius. Results show that RSU-based 20 V has 5% better performance when compared with a multi-objective algorithm.

Task replication

Figures 8, 9, 10, 11 show the comparison of performances of benchmark LC, multi-objective algorithm, and RSU-based models where task replication is applied. In Fig. 8, a comparison of a multi-objective algorithm is made with two models of RSU-based 20 V where straight task replication is applied with two and three replicas for every task. Multi-objective shows better performance when compared with all the models until the number of tasks reaches 25. After this RSU-based 20 V-2R shows better and very steady performance even the number of tasks is increased to 50. With exactly two replicas for every task, it shows a higher average execution time because of the smaller number of partitions for each job. It has a steady performance because due to task replication, there is a low probability of task failures. Overall RSU-base 20-2R

is showing 12% and 44% better performance compared to the multi-objective algorithm for 30 and 45 number of tasks, respectively.

In Fig. 9, a comparison of the performance of multi-objective and RSU replication-based models is shown where the number of tasks is fixed to 30 with varied cluster radius from 50 to 300 m. RSU model with exactly two replicas is better in performance and shows variations when the radius is more than 250; whereas, the RSU model with three replicas shows a steady performance even at a 300-m radius. It is because, with three replicas, it almost diminishes the expected location-based chances of failures. Results show that the RSU model has 23% better performance against the multi-objective algorithm.

Figures 10, 11 show the comparisons or performances between LC, multi-objective, and RSU models where location-based task replication is applied. In Fig. 10, the performance comparison of these algorithms is shown. In these experiments, the cluster radius is kept constant at 100 m and the number of tasks is varied from 10 to 50. RSU-based 20 V-LTR is showing the best performance over all the other models. It is because it has the highest number of volunteers and smart task replication that lower the average execution time and task failures chances, respectively. Figure 11 shows the impact of varied cluster radius on the performances of different models. RSU-based 20 V-LTR has better performance over other models because of the high number of volunteers. However, when the radius is increased to 300 m, it shows a slight increase in average execution time due to higher task replications caused by a higher probability of task failures. It is determined from the evaluation that RSU-based 20 V-LTR has 43% better performance when the number of tasks is exactly 30 and compared with a multi-objective algorithm. This is because when number of tasks is 30, the existing scheme has dropped some tasks and we assume that those tasks are locally computed which increased the average execution time. In RSU-based 20 V-LTR, with a greater number of volunteers, task replication is applied which decreases the job failures and improves the makespan. However, when average performance all number of tasks from 10 to 30 is taken this performance has 21% improvement against multi-objective algorithm.

After the analysis of results regarding the minimum number of volunteers, we conclude that a minimum of ten volunteers is necessary for effective task executions. Task execution capabilities and resource utilization can be enhanced using a greater number of volunteers. With 20 volunteers, 46% improvement is accomplished in average execution time and 30% improvement has been attained in the weight ratio for 50 tasks. With the increase in the number of volunteers, the computation capacity of the system is increased which increases the performance by reducing average execution time. With the increased resources in the form of volunteer

vehicles, the better performance is achieved. However, this better performance comes with the availability of resources and the willingness of volunteers. This performance can be suffered when there are fewer resources available and contrary to this, vehicles fulfilling the VCBV requirements might degrade the performance of other safety applications.

Conclusions and future work

In this paper, we proposed RSU-based VCBV to execute the jobs between the volunteer vehicles. We proposed the system model, job scheduling, and task replication algorithms. We performed the simulations using average execution time, and weight ratio parameters. The results show that the proposed scenario is better than the existing offloading scheme used in VEC for task execution. Our proposed scheme not only addresses the limiting functionality of remote servers in disastrous situations, but it also utilizes the surplus resources of vehicles.

This article covered one aspect of VCBV that is RSU-based task coordination. Future research directions include the proposition of advanced scheduling methods for multiple RSUs to avoid central entity failures. For incentivization in VCBV, a game-theoretic strategy for the participating nodes would be also considered. To diversify the autonomy and enhance the functionalities Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) loop can be used.

Acknowledgements One of the authors, Abdul Waheed, thankfully acknowledges the Higher Education Commission (HEC), Government of Pakistan, for awarding him the scholarship (Indigenous PhD Fellowship for 5000 Scholars Phase-II Batch-II) to pursue PhD and to carry out the research work.

Declarations

Conflict of interest There is no conflict of interest declared.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Al Ridhawi I, Otoum S, Aloqaily M, Jararweh Y, Baker T (2020) Providing secure and reliable communication for next generation networks in smart cities. *Sustain Cities Soc* 56:102080
2. Kanwal M, Malik AW, Rahman AU, Mahmood I, Shahzad M (2020) Sustainable vehicle-assisted edge computing for big data migration in smart cities. *IEEE Internet Things J* 7(3):1857–1871
3. Hasan R, Hasan R (2020) Towards designing a sustainable green smart city using bluetooth beacons. In: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), pp 1–6
4. Chahal M, Harit S, Mishra KK, Kumar A, Zheng Z (2017) A Survey on software-defined networking in vehicular ad hoc networks : challenges, applications and use cases. *Sustain Cities Soc* 35:830–840
5. Bibri SE (2018) The IoT for smart sustainable cities of the future : an analytical framework for sensor-based big data applications for environmental sustainability. *Sustain Cities Soc* 38:230–253
6. Yousefpour A et al (2019) All one needs to know about fog computing and related edge computing paradigms: a complete survey. *J Syst Archit* 98:289–330
7. Khan LU, Yaqoob I, Tran NH, Kazmi SMA, Dang TN, Hong CS (2020) Edge computing enabled smart cities: a comprehensive survey. *IEEE Internet Things J* 7:10200–10232
8. Jararweh Y, Otoum S, Al Ridhawi I (2020) “Trustworthy and sustainable smart city services at the edge. *Sustain Cities Soc* 62:102394
9. Zhang Q, Gui L, Hou F, Chen J, Zhu S, Tian F (2020) Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN. *IEEE Internet Things J* 7(4):3282–3299
10. Raza S, Wang S, Ahmed M, Anwar MR (2019) A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions. *Wirel Commun Mob Comput*. <https://doi.org/10.1155/2019/3159762>
11. Cisco (2015) Cisco fog computing solutions: unleash the power of the Internet of Things Connect, Cisco. https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf
12. Zahmatkesh H, Al-turjman F (2020) Fog computing for sustainable smart cities in the IoT era: caching techniques and enabling technologies an overview. *Sustain Cities Soc* 59:102139
13. Mouradian C, Naboulsi D, Yangui S, Glitho RH, Morrow MJ, Polakos PA (2018) A comprehensive survey on fog computing: state-of-the-art and research challenges. *IEEE Commun Surv Tutor* 20(1):416–464
14. Goudarzi M, Wu H, Palaniswami MS, Buyya R (2020) An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Trans Mob Comput* 20:1298–1311
15. Sunyaev A (2020) Fog and edge computing. *Internet computing*. Springer, Berlin, pp 237–264
16. Ning H, Li Y, Shi F, Yang LT (2020) Heterogeneous edge computing open platforms and tools for internet of things. *Futur Gener Comput Syst* 106:67–76
17. Huang C, Lu R, Choo KKR (2017) Vehicular fog computing: architecture, use case, and security and forensic challenges. *IEEE Commun Mag* 55(11):105–111
18. Peng X, Ota K, Dong M (2020) Multiattribute-based double auction toward resource allocation in vehicular fog computing. *IEEE Internet Things J* 7(4):3094–3103
19. Waheed A et al (2020) Volunteer computing in connected vehicles: opportunities and challenges. *IEEE Netw* 34(5):212–218
20. Sun J, Gu Q, Zheng T, Dong P, Qin Y (2019) Joint communication and computing resource allocation in vehicular edge computing. *Int J Distrib Sens Networks*. <https://doi.org/10.1177/1550147719837859>
21. Panadero J, De Armas J, Serra X, Manuel J (2018) Multi criteria biased randomized method for resource allocation in distributed systems: application in a volunteer computing system. *Futur Gener Comput Syst* 82:29–40

22. Wu H, Sun Y, Wolter K (2018) Energy-efficient decision making for mobile cloud offloading. *IEEE Trans Cloud Comput* 8(2):570–584
23. Nguyen QH, Dressler F (2020) A smartphone perspective on computation offloading: a survey. *Comput Commun* 159:133–154
24. Elazhary H (2019) “Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. *J Netw Comput Appl* 128:105–140
25. Wang P, Zheng Z, Di B, Song L (2019) Joint task assignment and resource allocation in the heterogeneous multi-layer mobile edge computing networks. *IEEE Internet Things J* 6(2):2872–2884
26. Sheng M, Dai Y, Liu J, Cheng N, Shen X, Yang Q (2020) Delay-aware computation offloading in NOMA MEC under differentiated uploading delay. *IEEE Trans Wirel Commun* 19(4):2813–2826
27. Qian L, Wu Y, Ouyang J, Shi Z, Lin B, Jia W (2020) Latency optimization for cellular assisted mobile edge computing via non-orthogonal multiple access. *IEEE Trans Veh Technol* 69(5):5494–5507
28. Hu P, Dhelim S, Ning H, Qiu T (2017) Survey on fog computing: architecture, key technologies, applications and open issues. *J Netw Comput Appl* 98:27–42
29. Zhou Z, Yu H, Xu C, Chang Z, Mumtaz S, Rodriguez J (2018) BEGIN: big data enabled energy-efficient vehicular edge computing. *IEEE Commun Mag* 56(12):82–89
30. Pu L, Chen X, Mao G, Xie Q, Xu J (2019) Chimera: an energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications. *IEEE Internet Things J* 6(1):84–99
31. Hou X, Li Y, Chen M, Wu D, Jin D, Chen S (2016) Vehicular fog computing: a viewpoint of vehicles as the infrastructures. *IEEE Trans Veh Technol* 65(6):3860–3873
32. Kai K, Cong W, Tao L (2016) Fog computing for vehicular Ad-hoc networks: paradigms, scenarios, and issues. *J China Univ Posts Telecommun* 23(2):56–65
33. Sorkhoh I, Ebrahimi D, Assi C, Sharafeddine S, Khabbaz M (2020) An infrastructure-assisted workload scheduling for computational resources exploitation in the fog-enabled vehicular network. *IEEE Internet Things J* 7(6):5021–5032
34. Nirmala SJ, Setlur AR, Singh HS, Khoriya S (2020) An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. *J Parallel Distrib Comput* 136:14–28
35. DaWang G (2007) Joshi, and Gregory Wornell, “Efficient task replication and management for adaptive fault tolerance in mobile grid environments.” *Futur Gener Comput Syst* 23(2):163–178
36. Jiang Z, Zhou S, Guo X, Niu Z (2018) Task replication for deadline-constrained vehicular cloud computing: optimal policy, performance analysis, and implications on road traffic. *IEEE Internet Things J* 5(1):93–107
37. Sun Y, Song J, Zhou S, Guo X, Niu Z (2018) Task replication for vehicular edge computing: a combinatorial multi-armed bandit based approach. In: *IEEE Global Communications Conference, GLOBECOM 2018-Proceedings*, pp 1–7
38. Li Z et al (2018) Fault-tolerant scheduling for scientific workflow with task replication method in cloud. In: *IoTBDs 2018-Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security*, pp 95–104
39. Wang Y, Duan X, Tian D, Lu G, Yu H (2013) Throughput and delay limits of 802.11p and its influence on highway capacity. *Proced Soc Behav Sci* 96:2096–2104
40. Rioul O, Magossi JC (2015) Shannon’s formula and Hartley’s rule: a mathematical coincidence? *AIP Conf Proc* 1641(1):105–112
41. Zhang J et al (2020) Prospective multiple antenna technologies for beyond 5G. *IEEE J Sel Areas Commun* 38(8):1637–1660
42. Firdaus M, Rhee K-H (2021) On blockchain-enhanced secure data storage and sharing in vehicular edge computing networks. *Appl Sci* 11(1):414
43. Zhao Q, Feng L, Zhao L, Li Z, Liang Y (2020) SatOpt partition: dividing throughput-stability region for IEEE 802.11 DCF networks. *IEEE Trans Veh Technol* 69(9):10278–10290
44. Henderson TR, Lacage M, Riley GF (2008) Network simulations with the ns-3 simulator. In: *Sigcomm’08*, p 527
45. Cui M, Zhong S, Li B, Chen X, Huang K (2020) Offloading autonomous driving services via edge computing. *IEEE Internet Things J* 7:10535–10547

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.