



A distributed task orchestration scheme in collaborative vehicular cloud edge networks

Shilpi Mittal¹ · Rajan Kumar Dudeja² · Rasmeet Singh Bali³ · Gagangeet Singh Aujla⁴ 

Received: 6 February 2022 / Accepted: 8 September 2022
© The Author(s) 2022

Abstract

The next generation vehicular networks would be expected to support a wide array of cutting edge applications concerning intelligent transportation system (ITS). Due to this reason, the scale and complexity of ITS-based compute-intensive tasks has exhibited a phenomenal increase and will continue to grow in future. Thus, a large quantity of data requiring different levels of processing is generated, that necessities the need of in-vehicle computational resources as well as collaboration from technologies like, cloud and edge computing. This has led to the development of paradigms such as vehicular cloud computing (VCC) and vehicular edge computing (VEC). Although VCC provides rich computing resources of the cloud servers to process tasks but it is affected due to long latency and instability of connections. In contrast, VEC provides compute resources closer to the data source to offset the relatively higher latency but the task requester should be able to perceive the computing and communication environment so as to allocate tasks effectively. Thus, it is essential to utilize both edge and cloud capabilities to create a collaborative cloud edge network that can cater to the demand of vehicular networks. A distributed task orchestration framework (DIOF) supporting a Vehicle-to-Vehicle based task orchestration scheme has been proposed that utilizes the vehicular movements along urban roads for creation of vehicular edges. The edge creation process utilizes an innovative light weight string processing algorithm based on hashing technique. The performance of DIOF has been evaluated based on extensive simulation by considering Chandigarh city road maps and the obtained results exhibit the satisfactory performance of DIOF for task orchestration.

✉ Gagangeet Singh Aujla
gagangeet.s.aujla@durham.ac.uk

¹ University Institute of Computing, Chandigarh University, Mohali, Punjab, India

² Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India

³ Department of Computer Science Engineering, Chandigarh University, Mohali, Punjab, India

⁴ Durham University, Durham, UK

Keywords Vehicular edge computing · Task orchestration · String hashing · Vehicle-to-vehicle communication

Mathematics Subject Classification 93A14 · 93A15

1 Introduction

The evolution of technologies in the domain of vehicular communications has become the key enabler for the rapid development of large number of applications based on intelligent transportation system (ITS). Coupled with emergence of smart vehicles, the scale and complexity of ITS-based computer-intensive tasks has exhibited a phenomenal increase and will continue to grow in future also. It is estimated that in next five years, the scale of in-vehicle equipment could reach nearly two billion resulting in 30 terabytes of data generation in a single day [1]. This will aggravate the requirement of the communication and processing capabilities related to the number of applications processing such a large volume of data. Thus, a modern vehicular network is required that is capable of addressing these issues so as to make it better equipped for handling future challenges especially with the advent of autonomous vehicles in near future. Although the autonomous vehicles of future are expected to produce large quantities of data that will require different levels of real time processing, the development of in-vehicle computational resources that are able to meet these processing requirements requires a huge effort. Highly dynamic topology and heterogeneity of computational capabilities coupled with varying traffic conditions are some other bottlenecks that will hinder the effective deployment of future vehicular applications. To address these issues, vehicular adhoc networks (VANETs) [2] are being integrated with different types of technologies such as Cloud Computing, Fog Computing and Edge Computing that has led to the development of paradigms designated as vehicular cloud computing (VCC) [3], vehicular fog computing (VFC) [4] and vehicular edge computing (VEC) [5] respectively.

One of the key requirements of these hybrid vehicular networks will be the realization of massive data transmission along with controllable latency. Traditional Vehicular Cloud Computing, facilitates execution of applications by allowing vehicles to offload their computational tasks to cloud data center by leveraging the rich computing and storage resources of the cloud servers to process tasks [6]. However, since most of these cloud data centers function through remote access, applications deployed through VCC are affected due to long latency time and instability of connections due to highly dynamic structure of VANETs. It also imposes a huge amount of stress on the load of network back-haul. VFC and VEC have been envisioned as two of the most promising technologies that keep in overcoming above issues. In VFC, the cloud service is pushed to the network edge called fog servers that act as fog gateways. The computation and storage resources are moved to proximity of users, which helps in reducing latency along with decreasing response time to a large extent. But, networks based on VFC often lead to large deployment cost due to relatively higher infrastructural requirements. Fog based computational models also have a number of limitations, such as better usability in geographically constrained environments, support for only

certain applications or requiring infrastructure-based centralized controls. The emergence of VEC has been considered to be the next developmental step to support the ever-growing computation-intensive and latency-sensitive services in ITS [7, 8].

In VEC, the data, applications and computing power is pushed away from the centralized network to extremes to offset the relatively higher latency in a remote cloud environment that sometimes affects Quality of Service. This helps to increase the computing capabilities by minimizing the need for remote data processing in the case of VCC, while also alleviating the requirement of additional infrastructure required in VFC. In VEC network, computational nodes can be deployed in cellular towers, Road-Side Units, as well as within connected vehicles. The proximity of data and computing process in VEC along with its implementation effectiveness has become the key reason for VEC becoming the preferred technology for task orchestration in vehicular environment. This is done by partitioning a given task into different jobs and allocating them for execution to neighboring vehicles that then process these jobs and transmit the result back to requester. Increasing number of connected vehicles and the limited communication resource poses one of the most critical challenge for deployment of VEC [9, 10]. This is because, VEC is generally implemented either through a statically aligned roadside infrastructure or through a group of dynamic vehicles moving along the road. In both the cases computation tasks are first offloaded to edge nodes, for processing and once the process is completed the result is transmitted back [5]. One important aspect that the task requester needs to consider is that it should be able to perceive the computing environment so as to allocate tasks effectively. This approach called the reactive approach incurs additional time overheads whenever a task allocation needs to be done. On the other hand in proactive approach, stored information about vehicles can be used for task allocation. Although, proactive approach requires additional storage, but it results in much more efficient vehicular edges.

In all these scenarios, any effective task orchestration scheme based on VEC requires a synchronised interaction between various computing and communication resources. Figure 1 describes the communication and computation process among vehicles through a VEC based framework. The figure depicts both the dynamic and static scenarios of vehicle edge creation. In both scenarios a vehicle can act as either a client vehicle or an edge server node. The client vehicle offloads task to nearest edge server for computing. In the case of dynamic scenario, vehicles rely on only V2V communication whereas in the case of static scenario, V2I come into the scene wherein a server edge is installed beside some RSUs for accomplishing task orchestration. Once a particular task is offloaded to edge server, it may compute the task at its own or distribute it among the vehicles. Task orchestration process deals with providing accurate response with minimum latency to the requester and is divided into task offloading, task computation and task execution management phases. Effective task computation in a vehicular environment can be considered to be the primary motive for deployment of VEC. In VEC, the task to be processed is first uploaded along with required data to the nearby edge server. This process is referred to as task offloading and edge servers then perform the computation based on the task requirements such as computation power, size of task offload and resource utilization. A properly designed task computation scheme helps in reducing latency and bandwidth usage by minimizing the need for long distance communications in conventional vehicular networks.

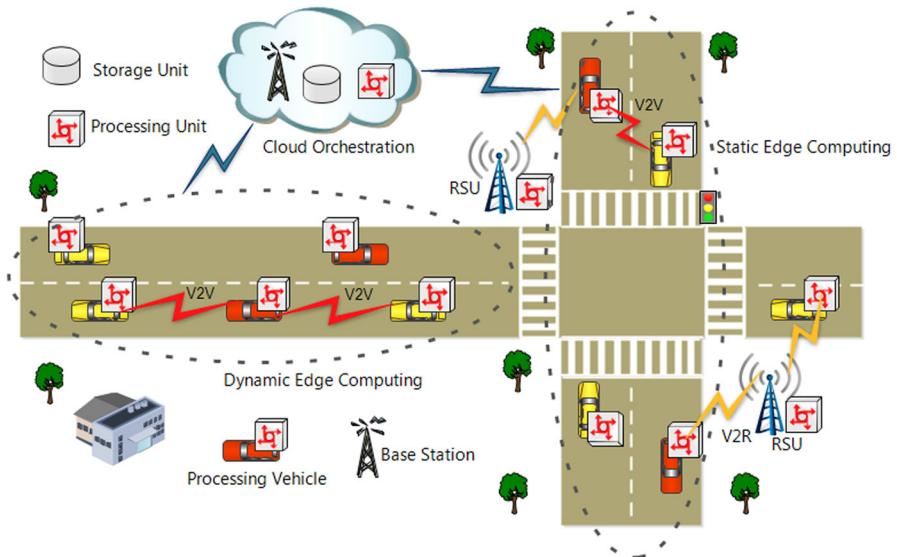


Fig. 1 Communication and computation scenario in vehicular edge computing

However, design of a suitable task computation scheme in VEC depends on factors such as channel allocation mechanism, edge nodes dynamics and optimality of task allocation. These processes are handled through task execution management module.

For a VEC, an optimal task orchestration depends on a number of diverse factors that need to be taken into consideration. A tightly coupled task reduces flexibility thereby resulting in higher latency for the allocated task. Channel allocation is also an important factor due to heterogeneous nature of vehicular communication systems. Since the vehicles or edge servers are equipped with multi-interfaces having different channel media such as conventional wireless or cellular communication efficient medium utilization also becomes a critical factor for VEC on account of this heterogeneity and dynamic vehicle density. Thirdly, a vehicular edge network requires an almost constant computational architecture for achieving efficient task computation. In spite of relatively high mobility of vehicles, an almost constant count of edge servers is required for managing the tasks effectively. Mobility of vehicles could also contribute to computation overheads on account of frequent handover in a VEC. The above factors also constrain the effective deployment of edge computing in VANETs since it entails requirement of additional infrastructure. This has resulted in limited applicability of edge based infrastructure in vehicular environment. Also processing large amount of data on resource-limited vehicle terminals is a huge challenge in VEC. Besides, frequent handovers due to short-distance communication range and high mobility also cause significant problem in VEC. Due to above reasons, recent research works for developing an edge network based on vehicular environment have focused towards minimal utilization of infrastructure.

1.1 Research contributions

In this direction, we have proposed a distributed task orchestration framework (DТОF) that is based on collaborative cloud edge network. An infrastructure less vehicular edge computing approach has been proposed where vehicles are used as network terminals that distribute task offloading locally. The proposed scheme creates an vehicular edge through vehicles that are moving on the roads and it also facilitates task orchestration and handling other edge functionalities through vehicle-to-vehicle communications. This is achieved by exploiting the predictable nature of vehicular traffic movement as well as discrete small time intervals for which the vehicles are stationary. The selection of edge nodes will be based on their destination and would be replicated on subsequent edge points whenever an edge needs to be created. This decentralized task orchestration approach will help in reducing data transmission latency and will put compute closer to where it is collected. The main contribution of the proposed work are as follows:

- A predictive distribution based edge vehicle identification scheme is proposed that assists in edge management and task orchestration.
- We propose a novel and light weight edge initialization model where the edge vehicles can seamlessly join and leave the edges.
- We propose procedure for task orchestration in proposed vehicular edge, which makes optimal decision on the allocation of communication and computing resources by observing the STATE of the VEC server.
- Finally, we conduct extensive simulations and results that the proposed DТОF scheme is efficient and highly reliable.

1.2 Organization

The remainder of this paper is organized as follows. Section 2 describes edge computing schemes along with some key proposals for implementing a VEC in vehicular network. Section 3 describes the proposed VEC Architecture and System Model for implementing Distributed Task Orchestration Scheme. The formulation and the corresponding solution for task orchestration and computation in VEC are discussed along with the proposed algorithms under urban scenarios is proposed in this section. Simulation results are presented to evaluate the performance achieved by our proposed methods in Sect. 4. Finally, Sect. 5 draws the conclusion.

2 Literature review

Hassan et al. [11] presented a survey on edge computing with respect to 5G and analysed some of the key services of edge computing in 5G networks. The key roles of edge computing for local storage and computation, local data storage and decision making, and local operations and security is also described. It also analysed the use of edge computing in technologies like Software Defined Network, Network Function Virtualization, Radio Access Technology etc. through 5G network. Raza et al. [12]

focused on issues related to extending role of edge computing in VANETs. Specifically, the key architectural concepts related to vehicular edge communication along with its applications and services is also analysed. Comparative analysis of cloud computing and edge computing from the perspective of vehicular communications is also presented. Dziauddin et al. [13] provided a comprehensive review of content caching and computation offloading in Vehicular Edge Computing. Some of the major challenges leading to bottlenecks in integration of VEC were also analysed and some of the key factors that can improve content delivery and content caching for future applications was also discussed. Dai et al. [14] proposed a Proof-of-Utility method to speedup block verification process from the perspective of security and privacy. Deep reinforcement learning (DRL) and Blockchain approaches are included into vehicular network with an aim of providing improved security for content caching and delivery. By exploiting the property of dynamic and decentralized one hop transactions of DRL and Blockchain, the proposed work introduced Blockchain at static vehicles and RSUs and DRL when the vehicles are mobile for content caching.

VEC is an important technology that can be used for reducing the task execution overheads for end users. Thus task orchestration functionalities such as task offloading and task computation are the major considerations in implementing efficient Vehicular Edge based Networks. A large number of task offloading and task computation approaches have been proposed in the literature. Huang et al. [1] proposed distributed reputation management based task management system to evaluate trustworthiness of cooperative vehicle by improving the problem of reputation in VEC. The proposed scheme focuses on the behaviour of users through a reward mechanism that achieves better task management. This system also addresses the issues of potential attacks in the network through a reputation management model. Huang et al. [15] investigated a parked vehicle edge computing approach where parked vehicles are deployed as edge nodes for implementing a task computation scheme. The parked vehicles that are a part of this edge are managed through edge servers. This approach provides an effective solution for task computations through reduced computation cost.

Stahpit et al. [16] proposed an algorithm to reduce the computation load in the network without the support of fog or cloud resources for real time applications. Instead of cloud or fog servers, smart cameras are utilized to compute task with the cooperation of its nearest cameras in the distribution manner. Pedestrians and mobile nodes are also used for task offloading to these cameras in this algorithm. Liu et al. [5] proposed a scheme to improve the quality of services while offloading tasks in VEC. The key objective of this scheme is to achieve delay optimization in different vehicular speed based road scenarios. A cost reduction based offloading algorithm is also proposed that specifically focuses on one-to-one and one-to-many task offloading strategies in the network. Dressler et al. [17] proposed a solution to manage distributed data in VEC using micro cloud based network model. A Macro-Micro-Cloud approach is developed for location-based services to increase the quality of service by reducing the communication cost in sparse networks when no neighbouring vehicle is available in network. This scheme is evaluated specifically for a V2V as well as Cellular Vehicular edge communication scenarios. Xu et al. [18] proposed a vehicle-2-everything (V2X) based computation offloading method that integrates V2X technology with task offloading. In this method, non-dominated sorting genetic algorithm is utilized

for load balancing based on determination of effective routing paths for task computation. Simple additive weighting and multiple criteria decision making strategies are used for optimizing task offloading in vehicular network.

Tran and Pompili [19] proposed a holistic approach for offloading task and allocation of resources in multi edge servers in a VEC. Task offloading and resource allocation problems are individually evaluated for achieving relatively closely optimal solutions through the application of heuristic algorithms for solving the NP hard algorithms. Li et al. [20] proposed a task offloading and computation model for enhancing performance in a vehicular network. The scheme considers multitask offloading along with different size of computation cost. A low complexity algorithm is also proposed to reduce the computation delay that is evaluated through task offloading from different lane scenarios and discrete speed of vehicles. Luo et al. [21] proposed an Edge-based Vehicular Content Distribution algorithm to improve the content distribution in VEC. Content and vehicle priority relationship are considered as key parameters in this algorithm. Zhang et al. [22] proposed algorithms to overcome the issue in the scenario of vehicles moving out from a network without completing its task. The algorithm considers two different vehicular transitions scenarios based on whether the mobility pattern for a vehicle is known or unknown. It is also implemented for highways, streets and real time road scenarios to evaluate its suitability for different vehicular conditions.

Zhang et al. [23] proposed a task offloading scheme based on multi-agent load balancing distribution for managing resources and computation in VEC. It also focuses performance benefits of balancing the computation load thereby highlighting its superior performance compared with traditional related schemes. Zhao et al. [24] proposed a handover scheme based on simple additive weight algorithm and greedy expectation maximization algorithm in VEC. These algorithms attempt to reduce handover overhead by reducing parameters such as computation cost, consumption of transmitted energy and task completion time. Cha et al. [7] proposed a distributed virtual edge creation scheme in a VEC environment. It utilizes free and available computing resources for edge management where as the edge nodes coordinate with each other to orchestrate the task by using distributed processing. Xue and An [25] proposed a task offloading scheme for a mobile edge computing based on Non-Orthogonal Multiple Access to improve the Quality of Service in VEC. This task offloading and resource allocation scheme works with multiple edge nodes to offload data to multiple edge servers at different locations at the same time. Lv et al. [26] proposed a task offloading scheme based on trajectory prediction in Vehicular network. This scheme provides handover services between adjacent RSU's when a moving vehicle is processing task while driving. Deep reinforcement learning techniques are used in this model for performing task offloading and computation management. The comparative analysis of existing work related to the proposed work has been shown in Table 1.

Although a number of schemes have been proposed that deal with task orchestration but majority of these schemes rely on infrastructure or static vehicles for performing task orchestration. Also, most of these schemes do not evaluate feasibility of VEC with respect to a real time task computation process. Therefore, this work proposes a Distributed Task Orchestration scheme that only utilizes vehicles on road and V2V communication for task orchestration in a vehicular edge environment. The next sec-

Table 1 Comparative analysis of literature review

References	Year	Objective	Communication		Scenario		
			V2V	V2R	Urban	Highway	Rural
[15]	2018	Optimize resource allocation, task distribution	✓	✓	✓	×	×
[16]	2019	Efficient task offloading, computational load balancing	✓	✓	✓	×	×
[18]	2019	Optimized task computation	✓	✓	✓	×	✓
[19]	2019	Optimized task offloading, resource allocation	✓	✓	✓	✓	✓
[20]	2020	Resource allocation for task computation	✓	✓	✓	✓	×
[22]	2020	Reduce latency during task offloading and computation	✓	✓	✓	✓	×
[23]	2021	Reduce latency and improve load balancing	✓	✓	✓	✓	×
[7]	2021	Idle resource utilization for task computation	✓	×	✓	×	×
[25]	2021	Efficient task offloading and resource allocation	✓	✓	✓	×	×
[26]	2021	Task offloading while mobility	✓	✓	✓	✓	×

tion describes the Architecture and System Model for the proposed task orchestration scheme.

3 Proposed framework

Task orchestration process is one of the critical issues for enabling effective computation in vehicular networks. An efficient task orchestration procedure will reduce latency and improve the applicability of these networks in different domains. The proposed dynamic task orchestration framework (DТОF) focuses on optimizing task computation in vehicular networks by implementing edge computing infrastructure as a back-haul. It considers a vehicular edge computing system (VECS) consisting of cloud servers that function as centralized computing facility. The cloud servers perform static vehicular and task allocation data management activities that is required for edge generation and alignment as well as for task orchestration which is then distributed to edges equipped with their own local storage through a primary edge vehicle.

In the proposed system model, if any vehicular application on board that is running on vehicles moving along urban roads needs additional resources for satisfying its computational requirements, it uses the available VECS infrastructure for meeting them. In DТОF, edge servers distribute data for computation among other co-operative edge computing devices and edge management is performed solely based on V2V communication. This V2V based edge management distinguishes DТОF from most of the contemporary VECS and also achieves effective task orchestration through these dynamically created edges. Following key assumptions have been made for describing the network architecture and system model of DТОF.

- The network is assumed to have a maximum vehicular density of N vehicles along a road segment and edge capacity of K vehicles where $K \leq N$ for all values of N and K .
- The maximum number of road segments connected to a particular road junction has been assumed to be four with each segment containing two bidirectional lanes.
- Every vehicle is assigned a unique V_ID that is used for identification of that vehicle as well as for managing its static information stored on the cloud.
- Edge network is controlled by a primary edge server called PE_i which is a vehicle residing in the same edge network and is responsible for task orchestration.
- The primary edge server is responsible for edge creation, edge alignment and other edge management tasks.
- Every vehicle is installed with a container for facilitating task orchestration.
- All vehicles have heterogeneous resource capabilities and edges are dynamically created whenever any task to be orchestrated has additional computational requirements that cant be fulfilled by the resources of a single vehicle.

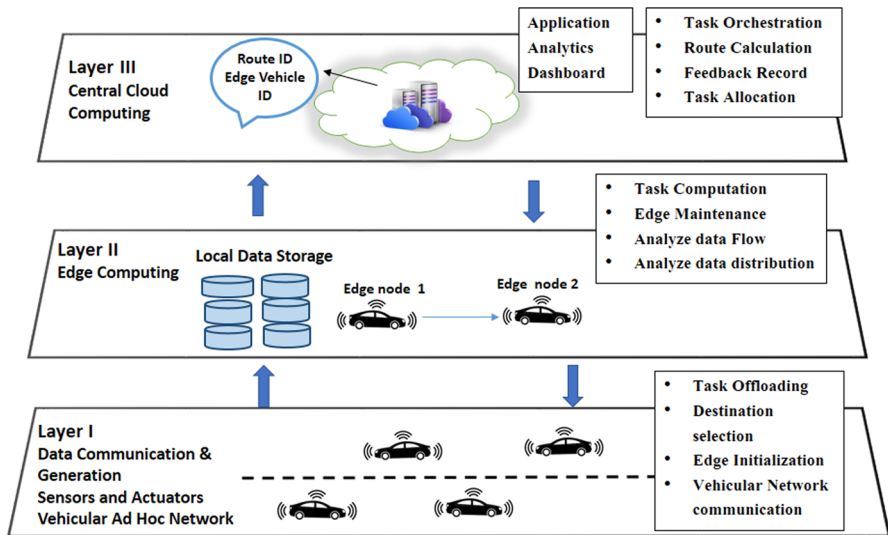


Fig. 2 Proposed layered architecture

3.1 Network architecture

Figure 2 describes the layered architecture, that consists of three layers used for task orchestration process in DTOF. Layer I designated as *Data Communication & Generation Layer* consists of vehicular nodes equipped with sensors and actuators that act as individual processing elements in VECS. In the proposed architecture, *Data Communication & Generation Layer* is primarily responsible for facilitating network functionalities related to route selection and initialization of vehicles for edge network. The sensing and transmission between central cloud and edge is also facilitated through this layer. Layer I functionalities also include edge network initialization and task offloading between edge network.

Layer II called *Edge Computing Layer* is mainly responsible for processes related to edge computation and edge maintenance. In Layer II, selected edge nodes analyze the data flow and distribute task among edge nodes while they are moving on the road. When an edge node leaves the edge or a new node joins the network it utilizes Layer II services. Layer III known as *Cloud Computation Layer* contains functionalities such as Vehicle Route Generation and user application interfaces are provided through this layer to VECS. It is also responsible for task orchestration by tracing the vehicles in edge network and the initialization of edge network at Layer I. The Layer III is also responsible for initiating actions when a task reaches a state that results in exception.

3.2 System model

DTOF performs its key functionalities in three different phases that are named as *Vehicle Route Initialization*, *Edge Alignment* and *Task Orchestration Phase* respectively. The dynamically organized vehicular edge network is formed on the basis of distance

to be travelled by vehicles through prior consideration of designated route that every vehicle needs to cover while it is moving towards its destination. This is achieved through a novel route initialization algorithm that has been proposed for computing the route to be taken by vehicles as they move towards their respective destinations. The Vehicle Route Initialization algorithm uses current location of each vehicle for performing edge management and task allocation through a lightweight procedure. DTOF also uses minimal cloud-edge interplay by considering both the cloud network as well as the dynamically created edge as independent functional units with very minimal interaction for task orchestration.

3.2.1 Vehicle route initialization phase

To initiate the edge formation process, vehicle route initialization is performed for every vehicle travelling along the city road. All road segments within a city have been grouped into sectors that are identified through a distinct sector number and road intersections at each sector that are called as road junctions. Every road junction is designated through a unique junction ID called J_{ID} . Each road segment has also been assigned a road ID called R_{ID} that has been considered to be the unique identifier for each road segment. The roads within a city can now be represented as an undirected graph $G(V,E)$ with each vertex in V designating a junction and all the edges in E indicating the various road segments that join these sectors in both directions.

To facilitate edge formation in the graph based model, the naming notation used for road segments considers the direction of moving vehicle along with the name of that road segment. The parameter R_{ID} is the identifier for every road segment within the city and is defined as a string of alphanumeric characters. The numeric value in the string represents J_{ID} , that represents the connecting junction between adjacent lanes. The alphabetic value associated with each lane is provided on the basis of direction in which the vehicle is moving along that road segment. For a road segment having two J_{ID} at either end, the lower valued J_{ID} preferred for assigning R_{ID} value. Figure 3 depicts these road segments names along with corresponding graph for these roads. As shown in the figure, character following the J_{ID} contains symbols U, D, L and R that designate the $UP, DOWN, LEFT$ or $RIGHT$ direction of the lane depending on whether the vehicles travel on North, South, East or West direction respectively along that segment. For example, road between the junctions 11 and 12 on South-North direction in Fig. 3 is represented by $11U$ and $11D$, respectively. Similarly, road between junctions 1 and 11 on East-West direction is represented by $11R$ and $11L$ in West-East direction.

Figure 3 also shows the complete route information for vehicles along the road. Here vehicle named $Vehicle_A$ has been shown to travel through a path that passes through junctions 21, 22, 23, 13, 3 respectively to reach its destination. The parameter R_I represents the concatenated string containing the $PATH_{ID}$ connecting the route taken as it will move towards its final destination. Thus for $Vehicle_A$ the value assigned to R_A is $21U22U23L13L3U$. Similarly, R_I is calculated for every vehicle I that is currently travelling on the city roads. These strings are then locally stored in every vehicle to indicate the complete path being travelled by that vehicle. These

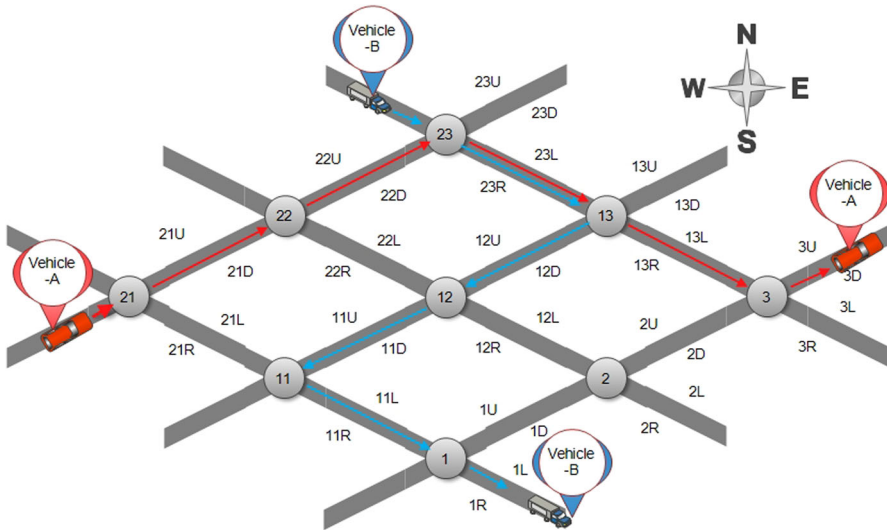


Fig. 3 Route identification in proposed framework

strings are also used in an efficient prefix string matching algorithm for determining vehicles travelling along the same route so as to include them in the edge network.

Table 2 depicts the important symbols used in the DTOF along with their brief descriptions. The Vehicle Route Setup is initiated as soon as some vehicle starts its journey by selecting its destination. The credentials of this vehicle along with its intended destination are transmitted for cloud based processing. The cloud first validates these credentials from its repository and also performs registration if setup operation is being performed for the first time for some vehicle. A centralized route selection algorithm is then invoked for computing the optimized route that is then transmitted back as a string to that vehicle.

Algorithm 1 explains the steps followed for implementing the process of Vehicle Route Initialization. The main step in the process of route initialization is to calculate the route from its source to destination by concatenating the R_I with calculated shortest PAT_{HID} for a vehicle. A shortest_Path function has been used to get the shortest route from source to destination. Further, to calculate the total distance between source and destination, A Find_Path_Length function has been defined that is based on shortest path computation process of Graphs. An additional function called Get_Path_ID has also been defined, which returns the PAT_{HID} for the next route to be travelled by the vehicle. The edge alignment process is then performed using this R_I based on the movement of vehicle along the identified path from source and destination.

3.2.2 Edge alignment phase

The proposed scheme uses an innovative string matching technique for edge formation and alignment that assists in seamless creation and efficient management of the formed edges. These on-demand dynamic edge's are established using a lightweight

Table 2 Notations of variables and functions used in DTOF

Symbol name	Description
V_{ID}	Vehicle ID
J_{ID}	Junction ID
R_{ID}	Road segment ID
R_I	Route ID
N	Total distance between source and destination
P	Primary vehicle
E	Edge vehicle
K	Prefix string length
I	Path ID based on the following junction towards destination
$PATH_{ID}$	Path ID
$Insert()$	Enter destination of smart vehicle
$State()$	Enters the state of vehicle
$Push()$	To get data from cloud to vehicle
$Find_Path_Length()$	Returns the total route length
$Get_Path_ID()$	Returns the path between one junction to next junction
$Shortest_Path()$	Returns the shortest Path between source and destination
$Concat()$	Merge the current R_I with next $PATH_{ID}$
$Push_Back()$	To store data from vehicle to cloud
$S[i]$	Prefix of i th character of the string corresponding to Primary edge
$H(S[i])$	Hash function of $S[i]$
$Transmit()$	Return the calculated Hash value
Γ	Random prime number
$Binary_Search()$	Binary search method
$Eligible()$	Check the eligibility of vehicle to join edge

string prefix matching procedure. This is based on maximizing the duration of connection among edge members by utilizing the predicted route of vehicle's as they move towards their respective destinations. The process of edge creation starts at road junctions when the vehicles are stationary so as to produce a homogeneous depiction about vehicular routes for improving efficiency of edge creation procedure while also producing effective edges.

The vehicle selection procedure relies on application of binary search on the lengths to find the longest common prefix. It is based on the premise that if a prefix of length K is common to both strings, then all prefixes with length smaller than K are also common to both. Similarly, if a prefix of length K is not common to both strings, then no prefix with length greater than K will be common to both. Because of this monotony, one half of the search space between Low and $High$ is discarded, every time a query for determining whether the length Mid is a common prefix or not, is performed. The value of median is computed as $Mid = (Low + High)/2$. Subsequently, string values of all the vehicles within transmission range is then compared with hashes of primary vehicle to determine the common prefix. This will correspond to largest

Algorithm 1 Vehicle-Route-Initialization Algorithm

Input: vehicle details, start location, destination

- 1: $State(V_i) \leftarrow ON$;
- 2: $Insert (Final_Destination)$;
- 3: $Push(V_i, V_i^{ID}, V_i^{Start}, V_i^{Final})$;
- 4: $Find_Shortest_Path(V_i^{ID}, V_i^{Start}, V_i^{Final})$
- 5: $N = Find_Path_Length(V_i)$;
- 6: $R_i \leftarrow NULL$;
- 7: $I = 0$;
- 8: $while(N \neq NULL)$
- 9: $PATH[I] = Get_Path_ID(V_i^{Start}, J_i, J_j, V_i^{Final})$;
- 10: $N --$;
- 11: $I ++$;
- 12: $end\ while$
- 13: **for** $X \leftarrow 0$ to I **do**
- 14: $Concat(R_i, PATH[X])$;
- 15: **end for**
- 16: $Push_Back(R_i, V_i^{ID}, V_i^{Start}, V_i^{Final})$;

common predicted duration for each edge vehicle. The hash computation is done using the following equation,

$$Hash(S[i]) = Hash(S[i - 1]) + S[i] * \Gamma^{(i-1)} \quad (1)$$

where, $S[i]$ denotes the prefix of first i characters of the string corresponding to primary edge whose hash is to be generated and Γ is a random prime number having length greater than a predefined value N . The value of N is dependent on the number of distinct characters used for designating R_p^i . In case of current implementation of DTOF, this value has been taken as 36 corresponding to 26 alphabets and 10 numeric digits used for creating R_i . The same procedure is used for computing hashes of all other member vehicles that are currently active in the edge,

$$\forall i \mid Hash(T[i]) = Hash(T[i - 1]) + T[i] * \Gamma^{(i-1)} \quad (2)$$

Once these hash values have been generated, binary search is then applied on the lengths of each vehicle, to determine their longest common prefix with the primary string S . In each iteration, the procedure selects a vehicle and the string (R_l) of length M for the computation, its and proceeds as follows:

We set the initial indexes as $Low = 0$, $High = \min(N, M) + 1$. In each iteration the invariant property that, Low will always indicate length of common prefix and $High$ will always indicate a length of uncommon prefix, is maintained. Since the efficiency of task orchestration process is dependent on effectiveness of string being processed, the edge creation step maintains linear time complexity. The preprocessing takes $O(N + M)$ corresponding to two linear for loops that calculates prefix hashes of the two strings. Binary search takes $O(\log(\min(N, M)))$, hence overall complexity is $O(N + M + \log(\min(N, M)))$ or just $O(N + M)$ as $\log(\min(N, M))$ is a lower order of magnitude than $O(N + M)$ and can be ignored in the asymptotically growth of the algorithm. Algorithm 2 describes the main steps involved in the process of edge

alignment process. In the first step, eligible vehicles are verified and added onto the edge based on their intended destination. The task orchestration is then performed using this edge based on vehicles moving along the same road segment.

Algorithm 2 Edge-Alignment

Input: Path-Rank, Junction-Details.

Notation: P=Primary vehicle, E=Edge Vehicles

```

1:  $H(S_P) = H(S_{P-1}) + \Gamma^{(P-1)}$ ;
2:  $N = Find\_Path\_Length(V_I)$ ;
3:  $Transmit(H(S_P))$ 
4: for each  $E \leftarrow 1$  to  $N$  do
5:    $H(S_E) = H(S_{E-1}) + \Gamma^{(E-1)}$ ;
6:    $M_E = Find\_Path\_Length(V_E)$ ;
7:    $LC P_K = Binary\_Search(0.Min(N, M_E);)$ 
8:    $Transmit(LC P_E)$ 
9: end for
10: for each  $E \leftarrow 1$  to  $N$  do
11:   if ( $Eligible(V_E) == 1$ ) then
12:      $Transmit(EDGE\_JOIN)$ 
13:   end if
14: end for

```

3.2.3 Task orchestration

To initiate the task orchestration process, a vehicle equipped with computational and caching capabilities is assigned a task that requires edge computational capabilities. This node then attains the role of primary vehicle ($\overline{P E_I}$) and starts the edge realignment process with its neighbouring nodes as soon as the vehicles reach its nearest junction. If an existing edge is available in its vicinity, the primary vehicle initiates the task orchestration process. In case an edge is not available, process of edge establishment is initiated through the edge alignment phase. After the edge member vehicles V'_p , are added onto the edge, task allocation is performed by selecting vehicles according to their Σ_I values. Finally, once the edges have been initialized, the vehicles start running and perform their respective task, till it is completed. The last phase of the framework relates to how a vehicle leaves the edge. As vehicles move towards their destinations, the value of R_I also keeps on reducing at each road junction. As this value reaches under the set threshold, it initiates the exit process of vehicle from the edge network. Before its exit, if a vehicle falls into the category of edge node, it transfers task computation information to the primary server vehicle of the edge network. After the task hands-off, required statistical information based on task orchestration performed by that vehicle is also updated on the cloud server and the vehicle then leaves the edge by transmitting an $EDGE_LEAVE$ message. However in case the exiting vehicle is a primary edge, it results in its task being aborted by transmitting a $TASK_ABORT$ message and the edge vehicles setting their STATUS flag to $IDLE$. Subsequently, the edge is reorganised at the next junction followed by next orchestration of the next task.

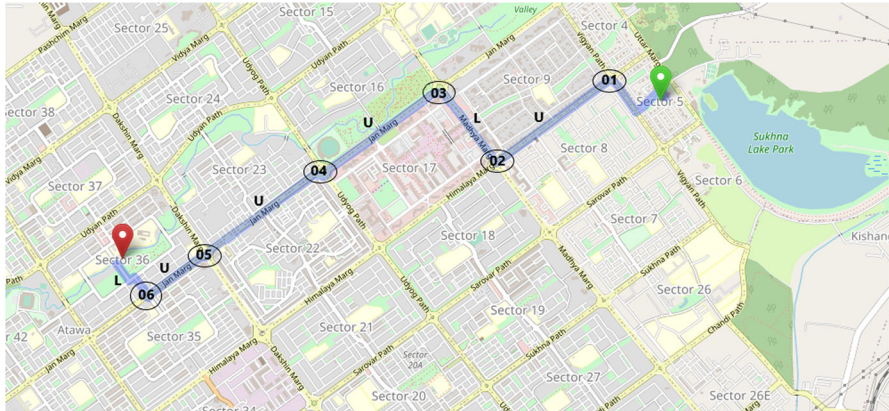


Fig. 4 Route-map from sector 5 to sector 36 Chandigarh, India

4 Implementation and simulation results

In order to evaluate performance of the proposed scheme, a series of experiments and simulations have been conducted using Network Simulator 3.29 (NS3) [27] and SUMO [28]. The evaluation results have specifically focused on issues related to edge initialization and task orchestration at different points along the road. Edge will be initialized during the time when the vehicles are waiting at road junctions while task orchestration will be done as they move towards their destinations. To evaluate performance, we have considered an urban area road map of Chandigarh city in India. As shown in Fig. 4 that has been created using OpenStreetMap [29] that shows the roadside scenario of Chandigarh depicting various points of traffic junctions along the road. These encircled points designate road junctions where edge network initialization is initiated whenever the vehicles stop at any of these points.

Table 3, depicts some sample route descriptions of roads in Chandigarh containing source, destination and R_I . For each route, a unique R_I is generated that depends on its corresponding route length and number of junctions that lie along that route. These R_I 's have been generated using Edge-Alignment Algorithm, that is involved at the centralized cloud server. Each R_I is obtained by concatenating all the J_{ID} that lie within that route. Since the distance between source to destination points of different routes are dependent on the number of junctions on that route, the length of R_I 's will vary. Therefore, For the routes shown in Table 3, the length of R_I 's varies from 9 characters that contains three junctions (01,02,07) to 27 characters containing 9 junctions (19,20,26,02,03,07,27,28,29).

Figure 4 shows the route directions for the case when any vehicle moves along a route having the source as Sukhna-Lake in Sector 5 while destination has been assumed to be Sector 36 that covers a total distance 6.9 km. To cover this route a vehicle will take at least 6 roundabout exits. These roundabout exits named as road junctions in our scheme indicate points where edge management is being performed so as to facilitate efficient edge management and task allocation. Based on the Route setup method, the computed Route ID of R1 is "01U02L03U04U05U06L" with length 18 character long

Table 3 Different routes of Chandigarh City

Route	Route description			Length (km)
	Source	Destination	Route ID (String)	
R1	SEC 5	SEC 22	"01L02U07U"	4.6
R2	SEC 5	SEC 36	"01U02L03U04U05U06L"	6.9
R3	SEC 11	SEC 43	"19U20U21U22U23R24U25L"	7.2
R4	SEC 5	SEC 43	"01U02U07U08U09U10L"	7.3
R5	SEC 26	SEC43	"33U34L35L07U08U09U10L"	7.8
R6	SEC 11	SEC 32	"19U20R26R02U03R07R27U28U29R"	8.3
R7	SEC 5	SEC 48	"11R12L13U14U15U16U17R18U"	11

Table 4 System parameters for simulation in NS3

Parameters	Values
Speed of vehicle	[50,80] km/h
Data rate (V2V communication)	27 Mbps
Arrival rate	Uniform:[0.165,0.33]/vehicles/s
Average number of vehicles/lane	[10,50]
Interval of sending BSM	1 s
Communication protocol	DSRC
Bandwidth	10 MHz
Computing power	Uniform:[0.1,1] GHz
Simulation time	120 s

as shown in Table 3. Similarly for each Route ID, a character string corresponding to its Route ID is generated, that is then provided to vehicles that move along these routes. To analyse edge performance in DTOF, NS3 based simulations are used by creating a network scenario corresponding to these *Seven* routes with vehicles arriving with differing arrival rates. The main parameters used in simulation are listed in Table 4.

The simulated model has considered data rate capacity of 27 Mbps with the bandwidth of 10 MHz. The speed of the vehicles has been varied from 50 to 80 km/h and vehicles arrives at random interval of [0.165,0.33] vehicles per seconds during travelling on route. Simulation follows DSRC protocol by including library "ns3/wifi-80211p-helper.h" along with "ns3/yans-wifi-helper.h". To support mobility in scenario "ns3/position-allocator.h" and "ns3/mobility-helper.h" libraries are used in NS3. The number of vehicles has been varied from a minimum value of 10 vehicles per lane that extends to 50 vehicles per lane. It is also assumed that each vehicle has its own computation power upto 1 GHz for task execution.

Figure 5 depicts a snapshot of simulation outcome when edge formulation process is being performed. The color of nodes depicts the type of node, with green color node in the middle of each lane functioning as primary edge node that communicates with other cooperative nodes in a distributed manner. Each node in Fig. 5 is identified by a node number so as to uniquely identify it and network communication between within

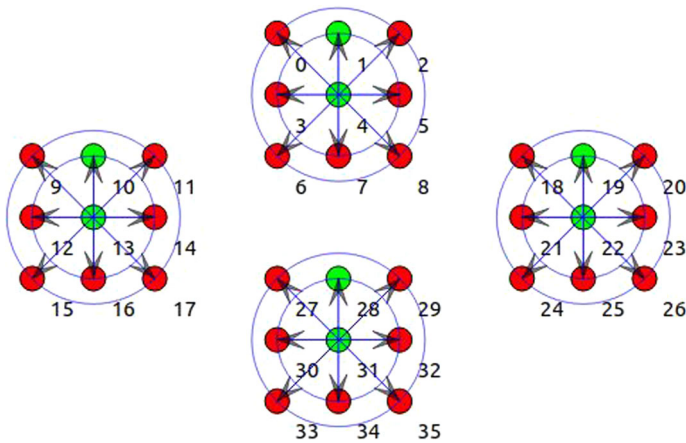


Fig. 5 Edge initialization at traffic junction

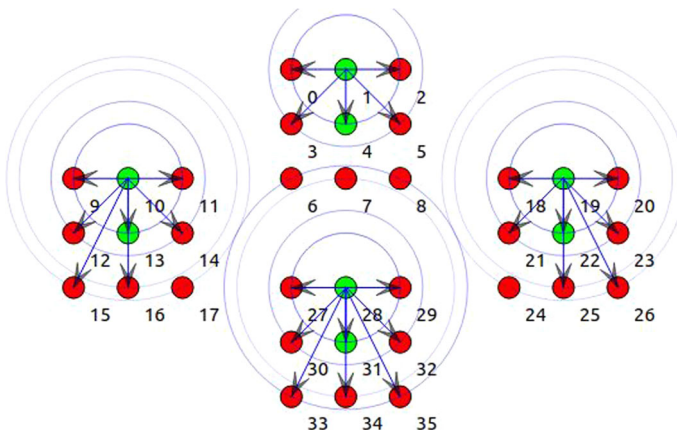


Fig. 6 On the way edge management

range nodes is achieved using message passing. The edge creation process concludes once all the eligible nodes join the edge. Now the primary vehicle performs task orchestration and edge network executes the task by utilizing all the moving vehicles that have joined the created edge. As the vehicular nodes are constrained by their transmission limit, there can be a possibility of a node not being in the transmission range while the edge formation is being performed. Figure 6 shows this condition in top most edge network where nodes 6, 7 and 8 are out of network coverage and similar condition is observed for nodes 17 and 24 in respective networks.

To evaluate the performance of the DTOF, parameters related to both communication and computing power have been considered. The evaluation has been done with respect to average number of vehicular node per lane that is defined as average of number of vehicles at different lanes. The key parameters used in our analysis are latency, packet throughput that are used for evaluating communication efficiency while

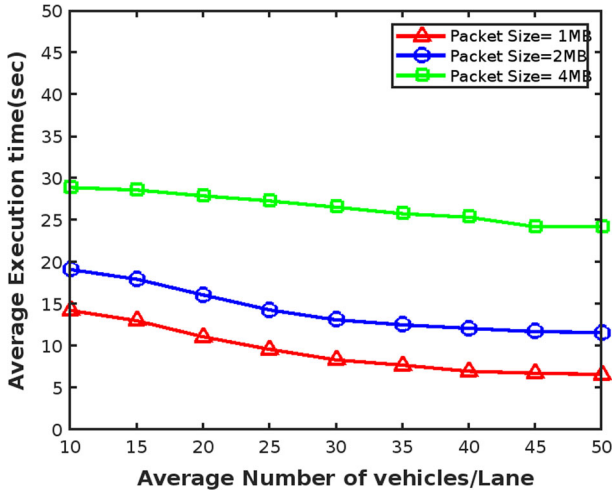


Fig. 7 Average execution time wrt average number of vehicles per lane

completion ratio, average execution time, number of failures and edge duration time have been used for analysing task computation effectiveness. These parameters are calculated from the generated trace files through multiple simulations for evaluating performance of proposed task orchestration scheme.

Figure 7 describes the impact of data size on average execution time for varying packet sizes of 1 MB, 2 MB and 4 MB. The graphs indicate that average execution time decreases as the number of vehicles are increased. Average execution time in DTOF exhibits a minor variation as the vehicle density is increased. This is due to the fact that as the number of vehicles is increased within an edge, higher number of nodes will be available for task orchestration that results in reduced execution time. Also this provides higher edge density thereby improving the execution rate of edges in DTOF. Figure 8 illustrates the variation in task completion ratio with respect to number of vehicles per lane. The value of task completion ratio initially shows a marginal increase when the average vehicles density increases and then attains an almost constant value for all three packet sizes. This is due to the fact that at smaller values of vehicular density, the task completion ratio also has a smaller value due to availability of limited computational capacity. Subsequently its value increases as the number of edge vehicles is increased due to higher computational capability of the created edges and reaches a saturation value from where it maintains a constant value.

Figure 9 depicts the performance of edge maintenance by evaluating the number of failures in terms of number of vehicles per lane. The obtained values from simulation indicate that the failure rate shows initial drop as the number of vehicles is increased and then reaches a minimum threshold value. The graph also shows that a larger task allocation results in higher task failure rate thereby resulting in relatively higher value of failure rate for large packets. Figure 10 shows the variation in edge duration time for different data sizes with respect to average number of edge nodes in the network. The graph shows that the time for which the edge is computing the allocated task increases

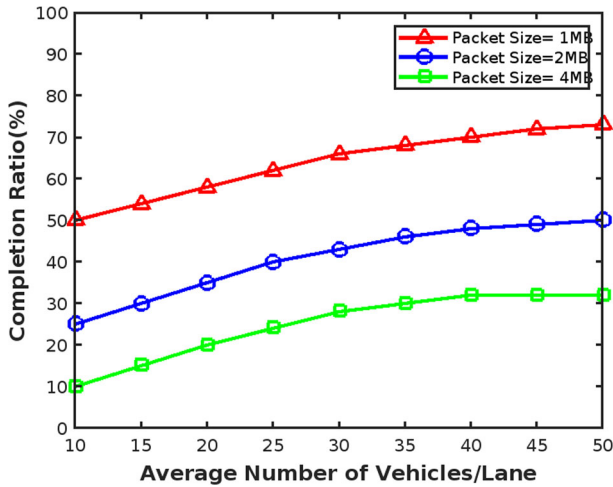


Fig. 8 Completion ratio wrt average number of vehicles per lane

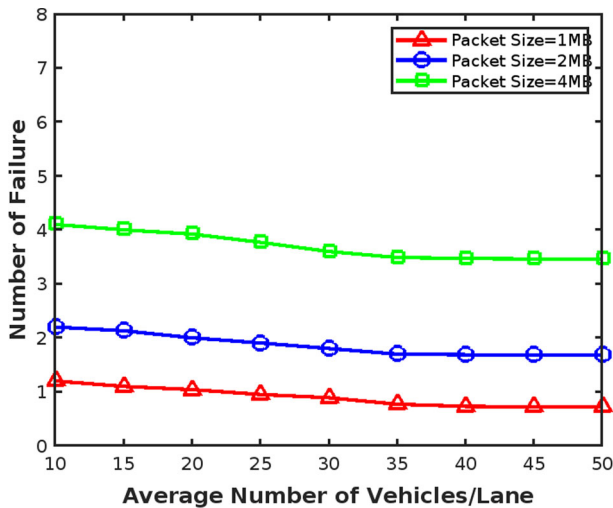


Fig. 9 Number of failures wrt average number of vehicles per lane

as with the size of the task because as the task size will be increased, more time will be taken to execute it with limited number of resources available in the edge network.

The performance of DTOF is compared with the existing Random (Rand) approach and Virtual edge scheme [7]. Rand selects the edges nodes randomly within the single-hop communication range whereas Virtual edge scheme has considered the link duration between vehicles and the available computational resources jointly while selecting edge nodes. Figures 11 and 12 show the variation in unexpected disconnections and edge duration in terms of average vehicular density per lane respectively. As compared with other two baselines, DTOF improves the stability of edge network by

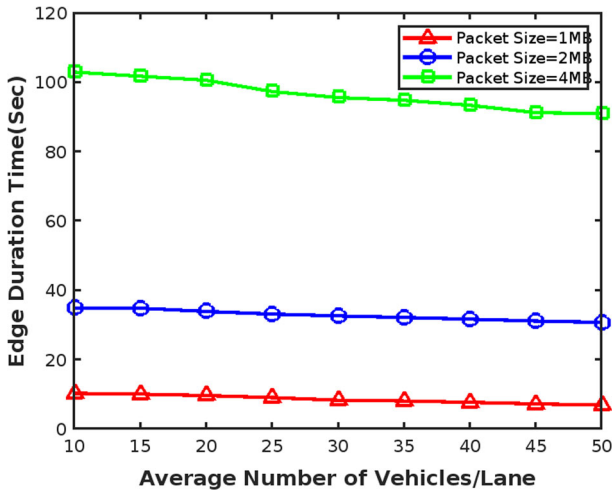


Fig. 10 Edge duration time wrt average number of vehicles per lane

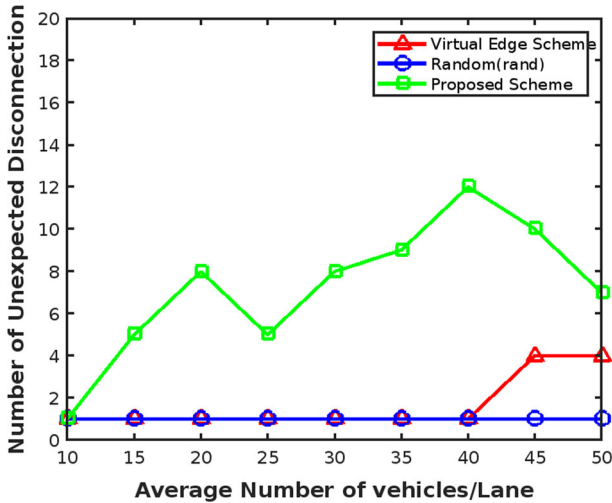


Fig. 11 Comparative Number of unexpected disconnections wrt average number vehicles per lane

considering the R_l and string match for measuring the distance for link-connection between the edge nodes. Therefore there is less chance to have link disconnection between edge nodes in DTOF. As a result, number of unexpected disconnections show relatively lower levels in DTOF. Therefore, it improves the edge stability and edge duration ensuring higher effectiveness of the created edges for task orchestration.

Figure 13 shows the impact on throughput of each edge network with the increasing size of edge network. The figure shows that throughput is exponentially increased as the number of nodes are increased. The reason for this behaviour is that in DTOF, computing nodes working in distributed manner are provided in the edge as the number of nodes are increased leading to improved packet throughput till it reaches a saturation

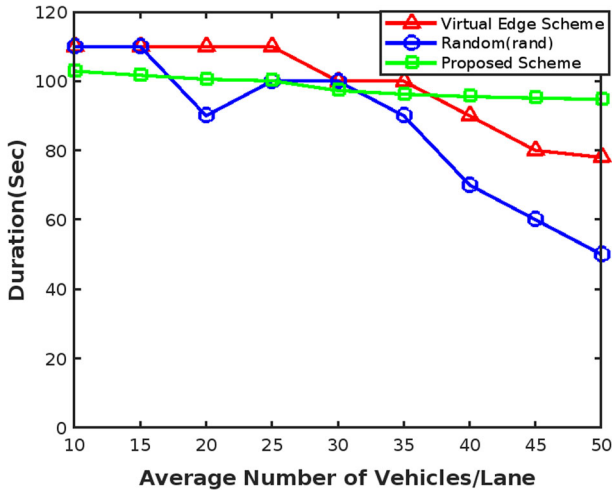


Fig. 12 Comparative duration of edge wrt average number of vehicles per lane

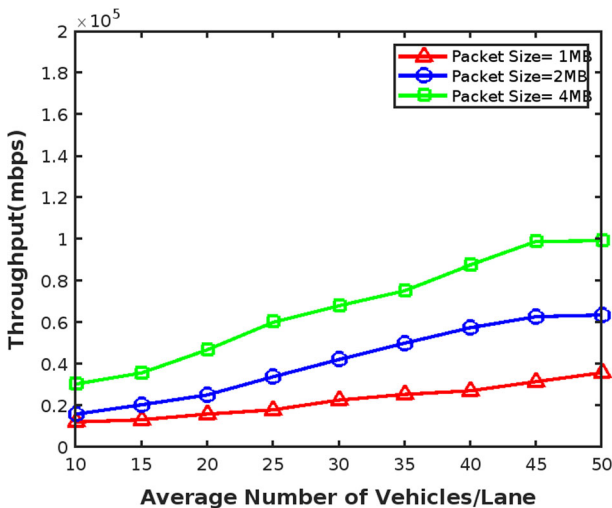


Fig. 13 Throughput wrt average number of vehicles per lane

value. Figure 14 illustrate variation in packet latency as a function of average number of nodes. The value of latency initially decreases by a small value for all the three packet sizes until it reaches a constant value. This is because of low overhead and light weight edge alignment process of DTOF that results in minimal data transmission. Thus, the available channel is utilized more efficiently thereby resulting in low packet latency in DTOF. The evaluated results exhibit that since the edge alignment process in DTOF is performed on stationary vehicles, the edge density remains constant while the vehicles are moving on the route. Also since the task orchestration procedure is performed seamlessly, the edge duration remains stable resulting in reduced overhead

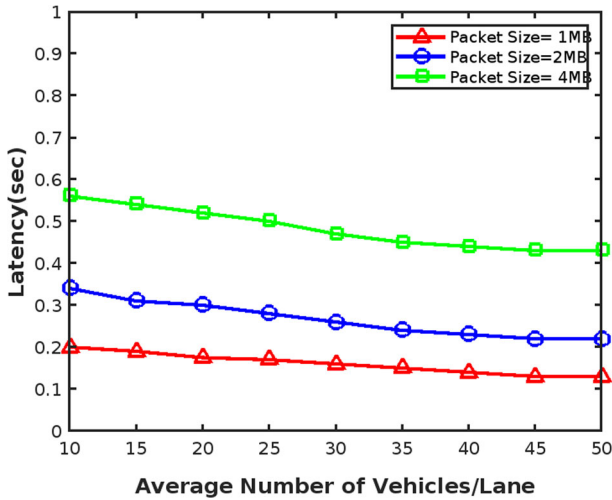


Fig. 14 Latency wrt average number of vehicles per lane

and minimum disconnections among the nodes in the edge network. This leads to relatively shorter data transmission time between edge nodes within an edge as well as higher transmission power utilization as the edge density is increased.

5 Conclusion

An effective vehicular edge computing based framework can be considered to an important technique for task orchestration through vehicular nodes. VEC also assists in better resource utilization without using any additional roadside infrastructure. The proposed distributive task orchestration framework attempts to achieve the above objectives through a collaborative edge formation scheme based on a light weight string matching algorithm. The obtained simulation results indicate better task completion ratio and average task execution time compared with existing schemes. In future, we will focus on optimizing the number of edge members within an edge according to requirements of different applications. We also extend our approach by considering intelligent based mechanisms to obtain the better resource utilization by enhancing the edge alignment process. We will also evaluate the performance of our scheme in different applications by conducting real-world experiments.

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. Huang X, Yu R, Kang J, Zhang Y (2017) Distributed reputation management for secure and efficient vehicular edge computing and networks. *IEEE Access* 5:25 408–25 420
2. Singh P, Bali RS, Kumar N, Das AK, Vinel A, Yang LT (2018) Secure healthcare data dissemination using vehicle relay networks. *IEEE Internet Things J* 5(5):3733–3746
3. Lee E, Lee EK, Gerla M, Oh SY (2014) Vehicular cloud networking: architecture and design principles. *IEEE Commun Mag* 52(2):148–155
4. 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
5. Liu L, Chen C, Pei Q, Maharjan S, Zhang Y (2019) Vehicular edge computing and networking: a survey. *Mob Netw Appl* 26:1145–1168
6. Tahmasebi M, Khayyambashi MR (2019) An efficient model for vehicular cloud computing with prioritizing computing resources. *Peer-to-Peer Netw Appl* 12(5):1466–1475
7. Cha N, Wu C, Yoshinaga T, Ji Y, Alvin Yau KL (2021) Virtual edge: exploring computation offloading in collaborative vehicular edge computing. *IEEE Access* 9:37739–37751
8. Singh A, Aujla GS, Bali RS (2021) Container-based load balancing for energy efficiency in software-defined edge computing environment. *Sustain Comput Inform Syst* 30:100463
9. Bitam S, Mellouk A, Zeadally S (2015) VANET-cloud: a generic cloud computing model for vehicular Ad Hoc networks. *IEEE Wirel Commun* 22(1):96–102
10. Garg D, Kaur A, Benslimane A, Bali RS, Kumar N, Tanwar S, Rodrigues JJ, Obaidat M (2021) Truclu: trust based clustering mechanism in software defined vehicular networks. In: *IEEE global communications conference (GLOBECOM)*. IEEE, pp 1–6
11. Hassan N, Yau KLA, Wu C (2019) Edge computing in 5G: a review. *IEEE Access* 7:127276–127289
12. 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* 2019
13. Dziauddin RA, Niyato D, Luong NC, Izhar MAM, Hadhari M, Daud S (2019) Computation offloading and content caching delivery in vehicular edge computing: a survey. *Comput Netw* 197(July):108228
14. Dai Y, Xu D, Zhang K, Maharjan S, Zhang Y (2020) Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks. *IEEE Trans Veh Technol* 69(4):4312–4324
15. Huang X, Yu R, Liu J, Shu L (2018) Parked vehicle edge computing: exploiting opportunistic resources for distributed mobile applications. *IEEE Access* 6:66649–66663
16. Sthapit S, Thompson J, Robertson NM, Hopgood JR (2019) Computational load balancing on the edge in absence of cloud and fog. *IEEE Trans Mob Comput* 18(7):1499–1512
17. Dressler F, Pannu GS, Hagenauer F, Gerla M, Higuchi T, Altintas O (2019) Virtual edge computing using vehicular micro clouds. In: *2019 International conference on computing, networking and communications, ICNC 2019*, pp 537–541
18. Xu X, Xue Y, Li X, Qi L, Wan S (2019) A computation offloading method for edge computing with vehicle-to-everything. *IEEE Access* 7:131-068–131-077
19. Tran TX, Pompili D (2019) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans Veh Technol* 68(1):856–868
20. Li H, Li X, Wang W (2020) Joint optimization of computation cost and delay for task offloading in vehicular fog networks. *Trans Emerg Telecommun Technol* 31(2):1–16
21. Luo Q, Li C, Luan TH, Shi W (2020) EdgeVCD: Intelligent algorithm-inspired content distribution in vehicular edge computing network. *IEEE Internet Things J* 7(6):5562–5579
22. Zhang X, Zhang J, Liu Z, Cui Q, Tao X, Wang S (2020) MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. *IEEE Trans Veh Technol* 69(3):3296–3309
23. Zhang Z, Li C, Peng SL, Pei X (2021) A new task offloading algorithm in edge computing. *Eurasip J Wirel Commun Netw*. <https://doi.org/10.1186/s13638-021-01895-6>
24. Zhao H, Zheng L, Li W, Zhou D, Li W (2021) Research on handover strategy based on greedy algorithm in vehicle edge computing. *Smart Innov Syst Technol* 190(September):59–65
25. Xue J, An Y (2021) Joint task offloading and resource allocation for multi-task multi-server NOMA-MEC networks. *IEEE Access* 9:16152–16163
26. Lv B, Yang C, Chen X, Yao Z, Yang J (2021) Task offloading and serving handover of vehicular edge computing networks based on trajectory prediction. *IEEE Access* 9:1–1

27. Carneiro G (2010) Ns-3: network simulator 3. In: UTM lab meeting April, vol 20, pp 4–5
28. Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, Lücken L, Rummel J, Wagner P, Wießner E (2018) Microscopic traffic simulation using sumo. In: 2018 21st International conference on intelligent transportation systems (ITSC). IEEE, pp 2575–2582
29. Haklay M, Weber P (2008) Openstreetmap: user-generated street maps. *IEEE Pervasive Comput* 7(4):12–18

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