Simulation Modelling Practice and Theory 98 (2020) 101968

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Low-latency vehicular edge: A vehicular infrastructure model for 5G

Venkatraman Balasubramanian[a], Safa Otoum[b], Moayad Aloqaily[b,c,*], Ismaeel Al Ridhawi[d], Yaser Jararweh[e]

[a] Arizona State University, AZ, USA
[b] Gnowit Inc., 7 Bayview Road, Ottawa, ON K1Y2C5, Canada
[c] Al Ain University of Science and Technology, UAE
[d] Department of Computer Science and Engineering, Kuwait College of Science and Technology, Kuwait
[e] Duquesne University, Pittsburgh, PA, USA

ARTICLE INFO

Keywords:
Vehicular model
Mobile edge computing
Vehicular service cloud
Vehicular cloud computing
5G

ABSTRACT

With 5G network services around the corner, vehicular cloud networks providing computation capabilities have taken precedence over traditional costly cloud solutions. However, with vehicular cloud computing, a variety of new challenges have grown. In this paper we propose an intra-vehicle resource sharing model to provide a range of cloud services such as on-demand entertainment and speech recognition for driver assistance. The proposed solution forms nearby low-latency Vehicular Service Clouds (VSC) on-the-fly as per the needs of vehicular users. Vehicles in parking lots or moving on the road collaborate and share their computation and storage resources to complete different vehicular service requests. We develop an incentive-based model that uses edge-based Road Side Units (RSU) to compose heterogeneous node resources and produce a usable resource that satisfies users' requests with minimal delays. Through proof of concept simulations, we compare our solution against traditional cloud solutions to showcase the effectiveness of adopting our proposed framework.

1. Introduction

Today's autonomous vehicles have revolutionized vehicular communication and the growth of vehicular technology [1,2]. The number of on-road vehicles have risen to an extent where statistics reveal that more 268 million vehicles are currently on the road and that the number has grown progressively over the years [3]. Moreover, with the increasing numbers in self-driving car tests, it has been predicted that by 2025 the US car market will account for over 6 billion dollars just for autonomous cars alone [4,5]. These autonomous smart vehicles run an excessive number of compute-intensive applications which require cloud assistance for data-processing and storage. After the development of the Cruise-AV car [6] by General Motors, it has been observed that smart vehicles have a potential of carrying computation units to serve other nearby vehicles.

Typically, requesting service from public clouds have disadvantages such as high Round trip time (RTT) and infrastructure costs. Such high RTT is not convenient for time-sensitive applications. In [7–9] the authors show that using remote clouds is not feasible for many services and applications. These services and applications are solely dependent on the time and place in which they need to be

* Corresponding author.
E-mail addresses: Vbalas11@asu.edu (V. Balasubramanian), Safa.Otoum@uOttawa.ca (S. Otoum), MAloqaily@ieee.org (M. Aloqaily), i.alridhawi@kcst.edu.kw (I. Al Ridhawi), Jararwehy@duq.edu (Y. Jararweh).

https://doi.org/10.1016/j.simpat.2019.101968
Received 29 April 2019; Received in revised form 8 July 2019; Accepted 12 August 2019
Available online 12 August 2019
1569-190X/

executed. Such place-bound services are best addressed closest to the user. Furthermore, these strict requirements motivate the need to integrate 5G mobile communication technologies with future vehicular networks. Thus, instead of requesting a service from the public cloud, smart vehicles could connect to each other forming an infrastructure for providing a wide range of services. To provide such services, smart vehicles must be equipped with computation and storage units. A problem that may arise is that these computation and storage units may not be fully utilized [10]. Hence, the problem of intra-node resource utilization must be considered, taking into consideration latency overhead.

In pursuit of this, we propose a framework that exploits the connected vehicle paradigm to provide low latency cloud services [11,12]. As investigated before in [13], the Edge Computing paradigm provides a suitable solution for integrating computation with the last-mile networks. However, it is of high significance to notice the fact that, these in-network deployments are costly and require infrastructural changes to an extent where legacy networks might prohibit such a change. Inevitably, with the growth of the connected vehicle technology and device to device applications, the attention should be moved to resource utilization while maintaining smooth and easy network integration. Additionally, most vehicles are not running computation intensive applications constantly. For instance, a speech recognition application that assists drivers such as Apple CarPlay [14] or an ambient sensing application that provides traffic statistics [15] will not be functioning while idling in a parking lot. In such scenarios, intra-node computation and storage can be used to provide services to near-by vehicles requesting assistance. Moreover, data caching and service composition in 5G networks using F2F can be also used in such scenarios [16,17].

1.1. Scenario

Road sensing applications [15] and multimedia processing applications [18] may behave better when data is acquired from nearby devices. For instance, when an on-demand entertainment request is generated from inside of a vehicle, it would require more time to process it through the cloud than if the processing is achieved in a nearby node. Furthermore, many subscription based overheads which require registration are needed when the processing is conducted at the cloud. On the contrary, if the requester discovers resources from nearby in-vehicle clients, hence, forming a local cloud, the result would lead to low latency services and reduced overheads, owing to a one-hop computation service. Moreover, this would provide a powerful platform for computation, data delivery, storage, and sensing. In pursuit of this, we propose a framework that enables cloud formation by collaborating between vehicular nodes, such that each vehicle becomes a cloud resource unit.

1.2. Methodology

As shown in Fig. 2, a request beacon is sent to a Road side Unit (RSU) for discovering nearby service vehicles that are willing to collaborate and share their resources to form a vehicular cloud. Once received, the diverse collection of vehicular resources (referred to as virtual resources) are composed into a usable vehicular infrastructure via service APIs [9]. A Vehicular Service Cloud (VSC) is a pool of individual in-vehicle servers with idle computational resources that collaborate with one-another in the vicinity to form a resource-rich pool. This low-cost computational solution is deployed over an environment where all nodes cooperatively maintain the network. Each vehicle can have multiple end-to-end connections (e.g., with the nearest RSU, cellular base station, nearby vehicular node, etc.). A key-enabler here is the protocol called Multipath TCP (MPTCP) [19] that assists in achieving increased application performance using multiple paths. Issues may arise in relation to vehicle registration with the RSU and mobility.

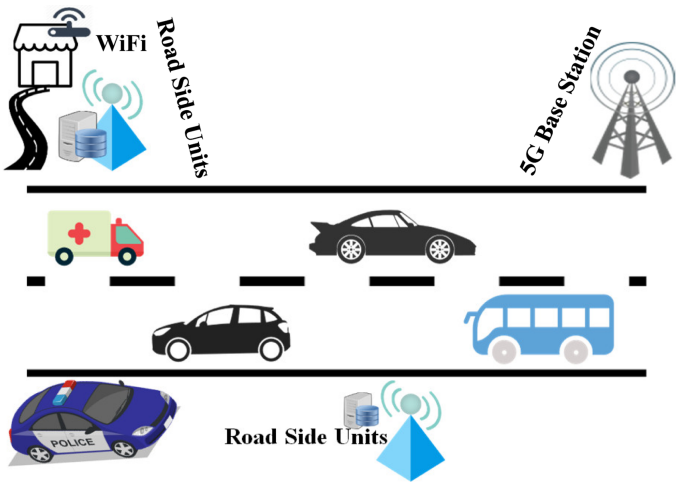

Fig. 1. Smart vehicle interaction with vehicular cloud entities in the environment.

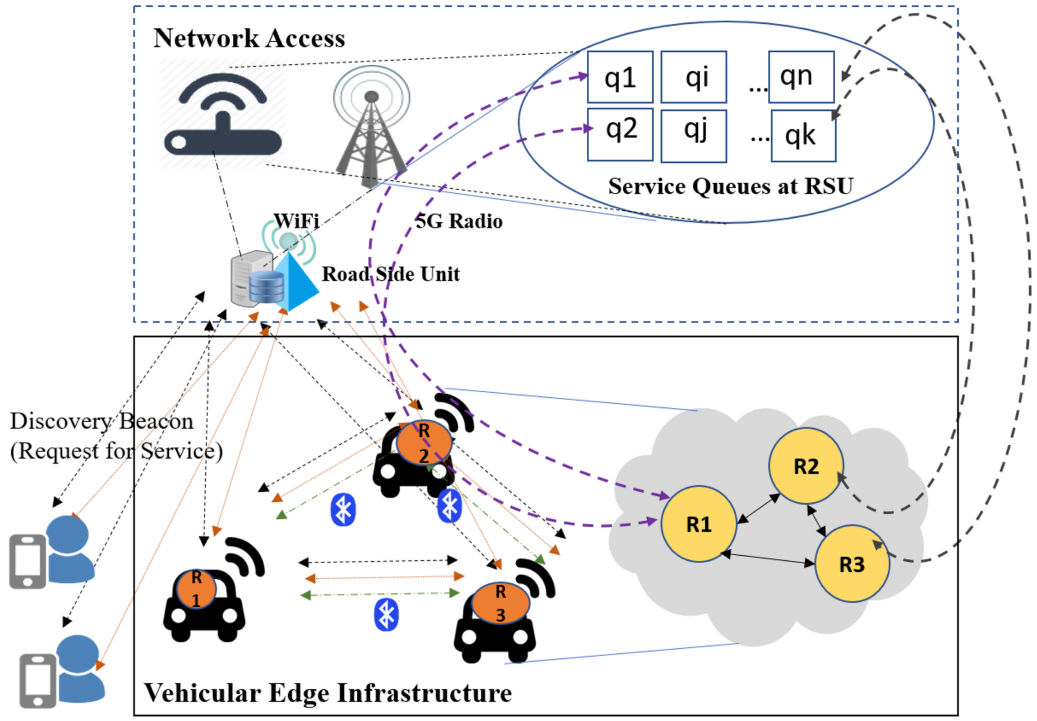

Fig. 2. Detailed overview of the interactions between vehicular cloud entities. Black dotted arrow represents external user to vehicle communication. Orange dotted arrow represents another interface for intra-infrastructure communication. Similarly, Green dotted arrow represents Bluetooth communication for close quarter scenarios (these are specific to MPTCP scenarios).

1.3. Vehicular cloud issues

- Due to high mobility, vehicular node links (i.e. the links between service providing nodes and the RSU) are highly affected. As seen in Fig. 1, nodes are able to connect to the closest base station and RSU. Owing to the stationary nature of the RSU, it has two links, one to the closest WiFi access point and another to the base station. Thus, the RSU acts as the operation point for resource discovery. In alignment with this, we propose a cloud composition framework that discovers share-able resources and creates communication links between the resources. In order to cater to heterogeneity of vehicles rather than focus on a single type of vehicle, we use this representation (i.e. Fig. 1). The representation is purely to demarcate how multiple vehicles can co-operate and communicate. The placement of police vehicle on the side of the road represents strategically placing surveillance and monitoring of RSU.
- Mobility of vehicular nodes may cause participating nodes to drop requests. Therefore, we test the proposed framework according to a specific highway scenario and compare it against related sate-of-the-art solutions.

To this end, our contribution mainly targets maintaining seamless service deliver and adequate end-to-end QoS.

1.4. Contributions

- We introduce a framework for vehicular service composition based on a peer-to-peer (P2P) strategy, where every node and its neighbours store the information needed to establish connectivity until the job processing is complete.
- We propose a vehicular edge architecture that addresses the congestion problem which imposes an application dependent policy for delay specific requests. Our objective is to minimize requests given a specific set of available nodes.
- We perform simulations to determine the average latency for service completion for different job lengths. Additionally, we conduct extensive evaluations for different vehicular scenarios to demonstrate the advantages of our framework. We observed over 67% successful offloading attempts and a latency of 0.7 s for end-to-end task completion.
- Finally, we perform comparisons against a state-of-the-art Vehicular Cloud model called AVARAC (Availability-Based Resource Allocation Scheme for Vehicular Cloud) [20] to evaluate our system. We observed over 10% increase in the rewards earned in our framework. Hence, this framework is a possible replacement for the vehicular cloud solutions in the next generation networks.

The rest of this paper is structured as follows. A brief discussion on state-of-the-art models and background information is provided in Section 2. In Section 3, we provide a preliminary overview of the proposed framework. Section 4 discusses the details in

regards to the RSU module. Discussions in regards to provider profit are considered in Section 5. In Section 6, we evaluate the performance of our proposed solution. Finally, Section 7 concludes this work and discusses potential future extensions.

2. Background and related work

2.1. Edge computing

Offloading to a remote data-center is costly and introduces overhead in terms of latency, in addition to infrastructure deployment which is experienced in traditional cloud solutions [21]. Fog technology has been introduces as one of the possible solutions for delay sensitivity especially for IoT services [22]. Researchers believe that in vehicular scenarios and 5G wires, “ultra dense edge devices including wireless Access Points, laptops, and base stations will be deployed with computation capacity as much as a server” [23]. It is widely agreed upon that the demands of sub-millisecond latency in 5G cannot be met by the cloud computing paradigm. This introduced the concept of Edge Computing. With edge computing, services that used to experience longer propagation delays are now moved closer to the end user, resulting in faster computation.

A major drawback of highly mobile environments is the fluctuating wireless connectivity, where users are moving from one region to another, resulting in loss of connections or disconnections during handover. Vehicular environments inherit these limitations. Hence, because of these limitations, having a remote location for task processing is not an appropriate solution. Edge Computing on the other hand has the benefit of having a one-hop computing environment such as a cloudlet, at a consumers disposal [24,25]. In vehicular environments, an RSU is a possible location that could behave as a location for brokering resources. In the next section we elaborate further why such a deployment can be beneficial with respect to a centralized cloud computing paradigm.

2.2. Vehicular infrastructure as a service

The resources housed inside each vehicle are share-able. Whenever applications are not making use of these resources, it renders the resources to be idle. Our motive through this article is to form a pool of such resources, where each vehicle volunteers a portion of its resources whenever needed. As shown in Fig. 2, once the initiating vehicle (also known as a the master vehicle (MV) sends a UDP beacon message with a request packet for resources, the RSU receives it and performs an overall look-out for vehicles that have registered for providing such a service. The request packet contains the amount of required resources for processing, the vehicle ID and an incentive for the new joining vehicle. This resource sharing mechanism involves potential benefits in terms of idle resources being used, and collaborating vehicles earning incentives. Once the RSU finds the required idle resources (slave-resources) from the vicinity, it multicasts the request packet to the specific nodes who have accepted to serve the request. These idle resources run a peer-to-peer application that composes heterogeneous resources together forming a usable compute resource. In doing so, the master-vehicle has the ability to form a Vehicular Service Cloud (VSC) used to execute different applications. This in essence confirms to the IaaS paradigm of Cloud Computing. Additionally, these vehicles have the capability to make use of multi-homing which a key-enabler of VSC.

2.3. Multipath TCP

In order to utilize the benefits of multiple paths that are simultaneously available, the Multipath TCP (MTCP) protocol was proposed in [26]. This protocol is an extension of TCP which allows applications to make use of multiple paths available in a network. It splits single TCP connections into sub-flows or streams that are transported dis-jointly via usable heterogeneous paths. It can co-exist with traditional TCP connections. In cases where the vehicles are MPTCP enabled, a typical control link established between them can lead to aggregated bandwidth usage. Further, since vehicles are mobile, having an anchor that manages control messaging makes it possible to maintain service continuity. Application of MPTCP on Vehicular Adhoc Networks (VANETs) is a well investigated topic in [19] and [27] it is beyond the scope of this article.

2.4. Related work

Authors in [28] envisioned a vehicular cloud involving cars. According to this approach, parking lots are considered to be the regions of cloud deployment. Authors do not consider high mobility or volatile scenarios as considered in this paper. Majorly, the consideration is given to travellers who leave their cars for a long time in the parking lot and travel for several days. Furthermore, node links are not the typical IEEE 802.11p DSRC as proposed in this paper. The framework proposed by the authors assume an Ethernet connection between the vehicles in the parking lot and a central server at the airport. In addition to not being realistic in terms of vehicle connections, a major drawback of the proposed framework is that it would fail in scenarios where the global picture of the vehicles is not observed. For tight scheduling of resources and to assign computational tasks to the various cars in the vehicular cloud, a fundamental prerequisite is to have an accurate picture of the number of vehicles that are expected to be present in the parking lot as a function of time.

In [29], Wang et al. suggest the usage of an edge computing framework in a mobile vehicular network. They consider an integral problem of heterogeneity between the edge computing architecture and challenges that may arise from the Internet of Vehicles (IoV) deployment. The authors proposed a collaborative framework called Vehicular Edge Computing framework (CVEC). The authors do not elaborate specifically on, nor target low latency-based communication, but focus more on how CVEC can support more scale-able

vehicular services and applications by both horizontal and vertical collaborations.

Zhao et al. present a bandwidth sensitive framework in [30]. In this work, the authors exploit the upload bandwidth of volunteers, which they call fog nodes, that form an edge-node-assisted real-time streaming P2P platform. According to this design, the cloud distributes task replicas to edge nodes, which respond to end-user requests. Authors use the public cloud as back-up to end-user demands only when the edge nodes cannot accommodate their requests. The authors make use of edge resources and decide on an optimal edge resource utilization strategy, mainly dealing with heterogeneity of edge nodes. Additionally, the job lengths are taken into consideration as the requests and bandwidth consumption are varied. In our work, we also consider the unequal workload lengths as well. Our proposal considers a completely mobile node environment that performs inter-node discovery, unlike the framework proposed by the authors.

In [10], Feng et al. investigate an autonomous vehicular edge scenario on the road with the aim of increasing the computational capabilities of vehicles in a distributed manner. The proposed framework manages the idle computational resources in vehicles in changing environments. Furthermore, the authors propose a work-flow to support the organization of vehicular edges. A job caching application is used, where neighboring vehicles supply requisite information. This work however, does not consider the latency-based constraints as has been done in our proposed framework.

The authors in [31] presented a hybrid vehicular cloud framework to provide increased computational capability of vehicles on the road through the use of resources from the cloud, road side units, and neighboring vehicles. The objective of the work is to provide excessive job processing and reduced communication costs. An online scheduling algorithm is developed which takes into consideration real-time and host requirements. To overcome issues that might arise in the construction of vehicular clouds, Choi et. al [32] proposed a multi-hop vehicular cloud construction protocol uses a connection time-based intermediate vehicle selection scheme to reduce the cloud failure probability of multi-hop vehicular clouds. Resources for a vehicular cloud are allocated based on the connection time between vehicles and the number of neighbor vehicles. In cases of connection failure and the leave of vehicular cloud member vehicles, a scheme for the reconstruction of vehicular clouds is provided. Simulation results conducted show that the proposed solution increases the service success ratio and decreases the service delay and the number of transmitted packets.

Yao et. al [33] considers vehicular fog services provided through vehicular fogs. Such services are formed on the fly by integrating a plethora of computing and storage resources of parked vehicles. Vehicular fog construction and service access is achieved according to a three layer framework. The first layer consists of the cloud and a fully trusted authority entity. The second layer consists of the vehicular fogs, in which each is composed of road side units and parked vehicles used as server vehicles. The third layer contains on-board units equipped on client vehicles. Mutual authentication between road side units and on-board units using a signature from the trusted authority is used to determine if a client has access to the vehicular fog service. Any malicious behavior detected is reported to the trusted authority to initiate the access revocation process. Monte Carlo simulations were conducted to evaluate the efficiency of the proposed mechanism in terms of reliability, security and latency.

Su et al. propose a D2D framework for vehicular social multimedia applications in [34]. The authors exploit a content delivery mechanism for parked and moving vehicles. The solution considers a solution where parked vehicles form a vehicular social community with moving vehicles passing on the road. This work differentiates from our work, where our solution focuses on vehicular cloud computation capabilities and is more driven towards providing low-latency infrastructure services.

3. Design overview

3.1. Preliminaries

Node Capacity - Each vehicle that is part of the vehicular cloud is known as a node. The capacity it carries to process a cloud request is referred to as Node Capacity. An initiating vehicle (as in a driver inside a vehicle) is referred to as a master vehicle (MV). While making a request, this MV generates a small XML packet carrying $< requestID, NodeID, \{QoS\ parameters\} >$. The RSU creates a local table of node-IDs and maps them to resource requirements of a request. Based on this request, a multi-cast message is sent. The composing agent/broker inside the RSU performs the main collaborative task as would be discussed in the next section.

Task profiling and Number of Participants - The job generation takes place in one node and is processed at a collection of nodes. This collection can only receive a portion of a job which we refer to as tasks. The method in which the job is divided is referred to as Task Profiling. This process happens inside the cloud initiating node (i.e. MV). Moreover, every task is embedded on the resources found by the RSU. Once the request submission process is over, RSU performs the fetching operation after which these nodes guarantee participation in the vehicular cloud. The nodes are called participants as they dedicate the idle resources to the master vehicle. As these participants are always willing to provide services to the master vehicle, we refer to them as slave-vehicles (SVs). The number of nodes required for processing is decided after the profiled tasks are available in the queue. When MV sends the beacon, the participant availability is sniffed by the RSU.

Request acceptance- Once the Road Side Unit (RSU) fetches the requisite resources, it runs the local composing agent (such as the distributed ad-hoc resource composition algorithm [9]) that creates a peer-to-peer link between the participants. This link sends data over the MPTCP layer and still benefits from the P2P communication. Furthermore, post-formation management and control message exchanges takes place to ensure the node is still actively participating in the cloud network. Every RSU has a specific buffer occupancy after which there needs to be a load balancing between RSUs for maintaining vehicular cloud QoS.

Link Utilization (Bandwidth)- The links are going to be fluctuating based on the mobility of the vehicle. However, each link has a specific capacity that is utilized based on the number of tasks being sent over the link. Our evaluation therefore also considers the link utilization with the number of requests accepted.

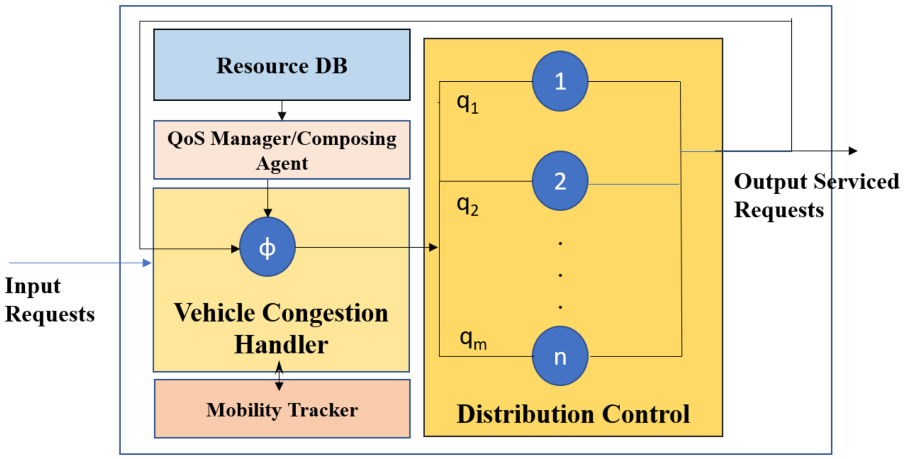

Fig. 3. Module interaction in the RSU.

3.2. Proposed RSU design

Fig. 3 shows the modular interaction of the proposed model that runs in the Road Side Unit.

- RSU Radio Module : This is the basic transmission and reception module running inside the RSU. Once a request for vehicular cloud formation takes place, the RSU sends a multi-cast message to a list of vehicles who are registered with it. These end-points act as the primary points for service provisioning.
- Distribution Control: This is a queuing model that caters to a time-sensitive queue depth. The queue length is prepared based on a predefined threshold. Typically, this is done to avoid a bloating condition. We do not modify the native queue and follow a FIFO M/M/1 strategy, but notify the local scheduler about the size of a shorter queue. Based on this, a request is offloaded to a shorter queue which enables maintaining low latency. The primary reason for such a maintenance scheme is to ensure short service times. The collection of queues from multiple interfaces ensures the resource data-base is updated with the latest information of the participants.
- Resource DB: This is the data-base of all the volunteers who are willing to collaborate with one another. The information needed to form a p2p connection and the network resources dedicated to a vehicular cloud composition in terms of bandwidth is stored here. It has a table for mapping each request packet that is registered with the RSU. At the time of initiation of a resource composition, the RSU decides to award incentives for those who complete the requests within a bounded period of time after the request is granted.

4. Characteristic features of the design

4.1. QoS manager and composing agent

The composing agent is the core component of this framework. Once requests are received by the RSU, it sends the multi-cast resource integration to a few select nodes that are ready to participate in the vehicular cloud network. Before the partitioned tasks are offloaded, the RSU first creates a logical link between the infrastructures following the P2P strategy previously deployed in [9]. The logical link formation process begins with the nodes submitting details in regards to node id, IP addresses, and available resources. After this, routing and management is done with the assistance of the information (key, value) in storage. A metafile is created, which uses sub-task information and resource information from the nodes. The metafile will contain the location of the source file used for processing jobs. Each node thus offers its idle resources, typically virtualized, to form an instance of an overlay network, where the computation can be carried forward. The term composition of resources is used in alignment with the cloud computing terminology of composition where heterogeneous resources are converted to usable form before offering it to a consumer.

Furthermore, the Composing Agent does the on-the-fly additions based on the available number of neighbouring nodes whenever a vehicle leaves the network. Every node already has the next hop information, such that, when a node leaves, another node is bound to join in without any new discovery procedure. That is, the network location of the node that comes in fresh is going to be the same as the node that leaves. What needs to be changed is a new label that the agent attaches to the new node. Every node that leaves is given a strict time-to-live stamp, based on which it has to find a new node or lose its credibility in a future service provisioning event. In this work we assume that nodes leave after establishing a new neighbour. Due to this, there are overheads with respect to this addition that comes in as an attach procedure similar to the Kademlia protocol [35] and its modification mentioned in [9].

The QoS Manager is aware of the velocity vector information obtained from the local RSU through sniffing, where it calculates the inter-vehicle velocity difference. This also gives information of the one-hop nodes apart from the node ID information gathered from the MV. The short XML message has QoS parameters which are evaluated here. Various parameters such as time for completion (TFC)

of a job, computation requirements, idle resources to share etc. are evaluated here. Based on this, incentive generation takes place. During conflict of resources, the vehicle congestion manager is given the control.

4.2. Vehicle congestion manager

The Congestion Manager module performs the final threshold allocation to the queues before dispatching the services. Whenever a conflict of resource utilization occurs or if a node that has become part of the service is no longer responding, then a threshold time for each node is invoked. It is important to do that due to the variations in the channels and inconsistencies at the time of queuing of sub-flows. The RSU pings the nearest base station or another RSU that is the next hop and determines if a new path can be established. The RSU has a multi-homing feature that enables sub-flows to a close by WiFi, as well as a cellular base station. However, as the congestion window scaling is different for both WiFi and cellular links, sub-flow management becomes integral. We adopt a ratio that is modeled based on prior work in [36], with the congestion window update having a multiplicative decrease (for packet loss in the path update *cwindow*/2) and additive increase (for each acknowledgment received in the update *cwindow* + Δ*cwindow*). The RSU sniffs the local area traffic for congestion (e.g. link shows high fluctuations resulting in transmission period overlapping) and all the flows are in this way catering to each others needs as the flow information is made global via the RSU.

4.3. Mobility tracker

Mobility Tracker is the agent that handles transient disconnections and maintains on-going sessions. The NodeID carries the information given in a route message format as $< IP, MAC >$. This shows the network location of the node. Essentially, the acknowledgement is set at 50 ms beyond which a new node is discovered by the RSU. At all points in time, the RSU finds a minimum of two nodes willing to participate in the vehicular network. For example, a highway is not going to be empty of vehicles and neither is an urban area such as downtown San Francisco [37]. The module examines the current location of the resource which has become part of the composition and updates the location in the database. The resource database is populated with the information gathered from this module. The tracking agent knows the current location of the provider.

For a given session, Node ID and a sequence number (request number being served) does not change. The pre-defined preferences enable selecting certain links based on the updates of the Mobility Tracker. Issues other than those pertaining to the composition, such as, different kinds of network triggered handovers and user triggered handovers are of importance are left for future work.

4.4. On-board NIC

This module runs on all nodes, more precisely, the vehicles and the RSUs. The nodes have a multi-homing capability, which enables connections to the closest base station and RSU. Owing to the stationary nature of the RSU, it has two links, one to the closest WiFi and another to the base station. This way, mobility tracking becomes far more easier. The on-board NIC makes an MP_Join call to add sub-flows. Owing to the heterogeneity of links, transport layer channels built over these links need to be monitored very closely. In order to distribute data among multiple transport connections, achieving a balanced workload among these connections is not a trivial task. Additionally, when moving from one access network to the other, there has to be seamless movements, which means the session is maintained despite the movement and change of link. All of these movements need to be monitored by the application. The scheduled jobs are aware of the specific nodes where they should be embedded. The Mobility Agent performs the flow check periodically and provides updates in regards to the status of the current node.

We consider a highway environment where velocity is fixed at 30–35 m/s, which is approximately within the legal road vehicle speed restrictions in the state of Arizona, USA. We also set the velocity for an urban environment to 20 m/s–25 m/s, where the movements are very much limited to 500 m. RSU's are deployed every 50 m. Furthermore, it does not happen often that a vehicle keeps moving at the same speed, however, for this work we restrict our movements within this speed limit. We evaluate how platooning affects the bandwidth in two velocity scenarios, namely, 25 m/s and 35 m/s. Furthermore, the scenarios are set up such that the vehicles are always under the purview of a road side unit.

5. Dynamics of latency and edge operator profit

5.1. QoS modelling: minimizing operator costs

Let us assume there is an RSU operator that behaves like an Infrastructure Operator which controls the resources allocated to each vehicle involved in providing a service. Suppose there are I vehicles between which resources are distributed in stages (formation of composition [9]). For example, a composition is created between a select few vehicles. Let us assume this happens in n stages. Now at the end of n stages, we determine that we need j more resources to satisfy QoS bound on latency. That is, we impose a penalty cost $C(j)$ (in terms of wait-time during on-the-fly addition), here the function increases with respect to j. Consider the profit earned by the edge operator to be P_i. Let us say $\kappa_n(i)$, is the minimal cost incurred while service composition is yet to be formed, from Ross [38], we have:

$$\kappa_n(i) = min_{y \geq 0}\{y + Pr(y)\kappa_{n-1}(i-1) + [1 - Pr(y)]\kappa_{n-1}(i)\} \quad (1)$$

Table 1, shows the symbols and their definitions. Here, the probability of successfully forming a composition is given as $Pr(y)$.

Table 1
Symbols and notations.

Symbols	Definition
$C(j)$	Penalty cost for j extra resource
I	JNumber of vehicles
n	Number of stages
P_i	Operator Profit
$\kappa_n(i)$	Cost incurred at ith stage
$Pr(y)$	Probability of successful allocation of y set of resources
j	Extra resource requirement
M_c	time to complete the processing of all jobs
B_s	time taken for relieving the resource
$1/\lambda_n$	Mean service time

Moreover, y is the allocated set of resources. The boundary condition is given as:

$$\kappa_o(i) = C(i) \tag{2}$$

So, as the value of i increases, $\kappa_n(i)$ also increases.

5.2. QoS modelling: bounded latency

Minimizing the duration of congestion avoidance is based on congestion avoidance cycle time and the number of jobs transmitted during the congestion avoidance cycle. This is the time taken to relieve a resource from its busy time. Assuming there are i jobs attempting to transmit at time δt, and knowing that the service time for n jobs is exponentially distributed with mean $1/\lambda_n$, then for i jobs' transmission will be completed in $\lambda_i \delta t + o(\delta t)$. In this period, assuming the time to complete the processing of all jobs is M_c and B_s is the time taken for relieving the resource,then no other job is buffered until the processing of the job is complete. This is also known as idle time. Hence, our objective Z becomes:

$$Z = min\{E[M_c - B_s]\} \tag{3}$$

That is, just by minimizing the time the processors are idle will minimize the overall time taken to process. This in turn increases the expected profit of the operator. For a jobs less than time t:

$$E[Profit] = P_i Pr(X_i < t) \tag{4}$$

$$= P_i(1 - exp(-\lambda_i t)) \tag{5}$$

$$= \lambda_i P_i \frac{1 - exp(-\lambda_i t)}{\lambda_i} \tag{6}$$

$$E[Profit] = P_i Pr(X_i < t) \tag{7}$$

$$E[Profit] = P_i Pr(X_i < t) \tag{8}$$

From Ross [38] we know that, $E[X]$ is given as:

$$E[X] = \int_0^\infty p_x(x)dx \tag{9}$$

Algorithm 1 summarizes the method used to select a set of vehicular nodes for task execution.

6. Performance analysis

Simulations are conducted using Omnet++ with the Veins Simulator [39] along with SUMO [40]. We generate a random map of Luxembourg and manually generate a topology with RSU and base station channels. The network connections between vehicles are purely an IEEE 802.11p DSRC framework. We use a carrier frequency of 5.9GHz band. The default transmission happens at 20 mW power and maximum radius is set to 500 m in which mobility traffic is simulated. We follow a generic fading model and a random service channel with each channel having equal selection probability. For our vehicular movement, we specify a destination. During the movement the vehicle operator activates an application for taking pictures of nearby objects and periodically shuts down. The application executed creates a set of tasks that are between the size of 100 KB and 1GB.

The request partitions are context-free and are submitted to the RSU once the nearby discovery is carried out and a set of vehicular resources are chosen. We generate requests following a Poisson arrival with each interval set to 10 s. This way the RSU is kept busy at a very high pace. We toggle the value to 50 s in the simulation, however, that does not produce any recognizable change in the processing time.

Each Vehicle processes the job at a rate that we $R \sim U(1, 5)$. We chose these parameters specific to our scenario, such that they can be varied based on changing applications. We perform an evaluation based on a non-vehicular cloud scenario, where local

procedure LABEL RESOURCES
  Input: $J_{un} = \{j_i, j_{i+1}, ., j_n\}$
  Input: $D = \{l_s^n; \forall n \in N\}$
  Output: list of nodes selected for task execution
  D_i = **UDP beacon sent**
  X_j= Resource Sample Space. ▷ Initiate P2P algorithm
  J = **TaskAccept(**$J_u n$**)** ▷ J resource to task mapping
  for j_n in J **do**
    while D not empty and J is not empty **do**
      if ScoreCal(j_n) $\leq L_r^n$ **then**
        $n \leftarrow j_n$ ▷ mapping task j_n to node n
      else
        Go to next node in the list D
      end if
    end while
  end for
end procedure

Algorithm 1. Vehicular Cloud Service Composition.

Table 2
Simulation parameters.

Serial	Parameters	Values
1	Requests	U(1,5)
2	Job Interval	50 s
3	Max.Node Density	1000
4	Smallest Task Size	100KB
4	Largest Task Size	1GB
5	Percentage Resource Share/node	10%

Table 3
Scenarios for evaluation within Omnett++.

Scenario No.	Criteria
1	Local Processing
2	Remote Processing (Urban or Highway setting)
3	Event-Driven Processing (Adding nodes on the fly)

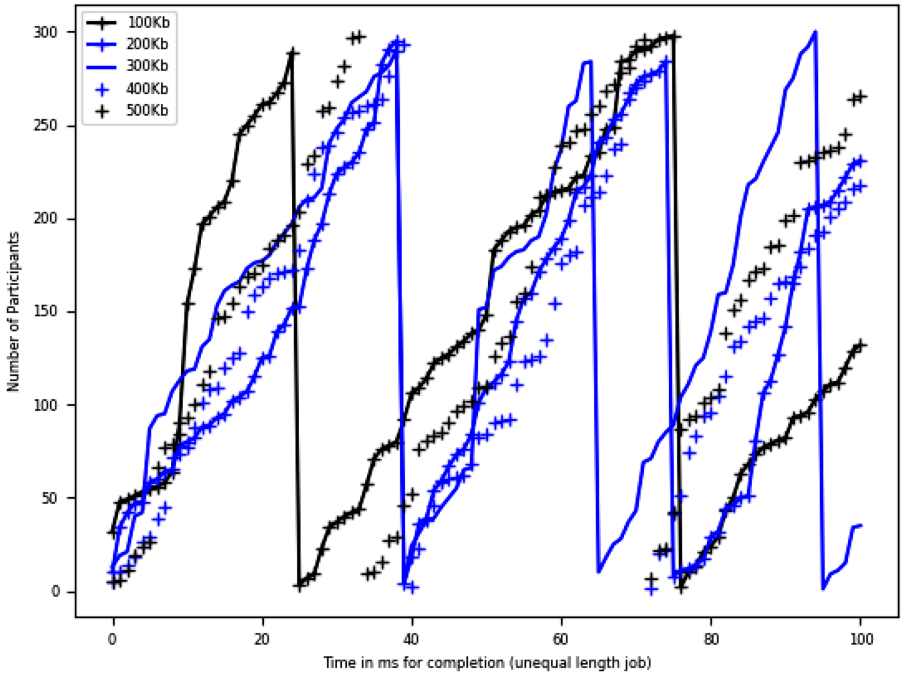

Fig. 4. Time needed to complete unequal length workloads.

processing is given priority with out any offloading, then compare it with an offloading approach. Processing locally means the application utilizes only the intra-node capabilities. Here, there is no job assignment strategy, a round-robin approach is followed by default. While offloading, we ensure a shortest path to the computation node with the highest resource (based on availability). In each topology, we vary the number of nodes from 10 to 1000 nodes. Dynamically based on arrivals, the tasks are embedded onto the nodes. Counter-intuitively, in cases where the number of requests is more than the available nodes, we provide an embedding mechanism by adding new-resources on-the-go to the existing infrastructure. Following the definition of active vehicles present in the area from Sichitiu and Kihl [41], we refer to such vehicles as the ones participating in the cloud network. We provide an average result obtained from 10 simulation runs. Vehicle velocity is kept between 20 and 30 m/s in urban areas and 45 m/s in highway scenarios. Table 2 summarizes the simulation parameters.

From Table 3, it can be seen that we provide three scenarios to evaluate our solution. First two scenarios are intended to be more generic evaluations, with the local processing scenario taking into consideration only intra-node resources and the remote processing scenario following the discovery of nodes nearby. The third scenario is based on adding new resources on the fly. Hence, the third scenario is *event-driven*.

Fig. 4 shows the time needed to complete unequal length jobs given the number of nodes participants. Fig. 5 shows the number of participants in a VSC from the available nodes in the network.

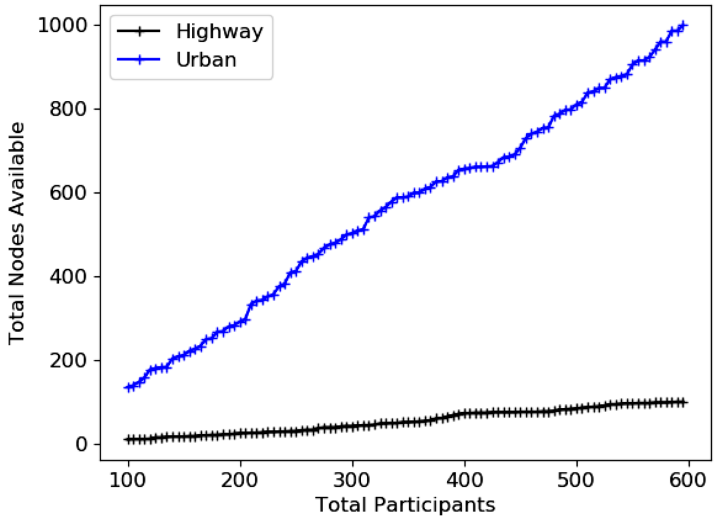

Fig. 5. Number of participants in a VSC given the total nodes available.

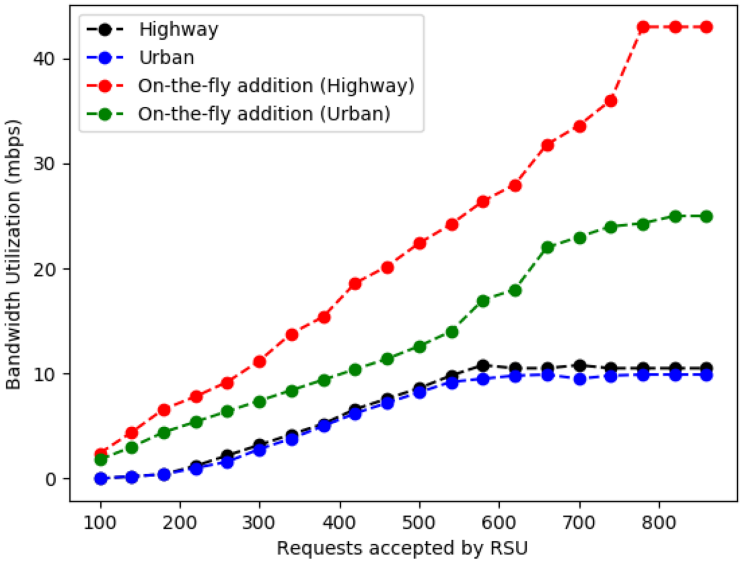

Fig. 6. Bandwidth utilization.

Fig. 6 shows Bandwidth Utilization. Results show a rather weird behavior in the event-driven scenario. The longer time for saturation in the highways is attributed to the wait times where the channel is still kept active due to a new joiner taking the same link as the node which left the cloud composition.

Although, addition of nodes on the fly shows feasibility, it is not recommended, as the time taken for a node to join in can play a crucial role, see Fig. 7. Essentially, if requests are being serviced at a consistent pace, there is no need for a node to join. However, in case of heavy mobility, addition of nodes can be beneficial as it does not need a new neighbour discovery to take place at the same time it guarantees job completion. Figs. 8 and 5 show the general overview of the number of tasks generated.

The overhead for offloading a job is the time needed to form a cloud, in addition to the neighbour discovery time. However, with the increase in node density, it is observed in Fig. 9 that after a certain point (i.e. 300 nodes) latency reaches a constant value. Typically, the values are higher in highways where there is an extra overhead for waiting for nodes to join as the density is comparatively low. Additionally, it is quite intuitive that on-the-fly addition would incur an extra time, hence, we avoid showing intuitive plots.

Fig. 10 shows the variety of workloads that we consider. The smallest task size is kept at 100Kb for equal length tasks. However, for unequal length task, we vary the job length from 100Kb to 200Kb, then to 300Kb at a random interval and arrange them based on the plot shown in Fig. 4. As expected, the behavior shows a very widely spaced time for completion as opposed to a constantly increasing time for equal length tasks.

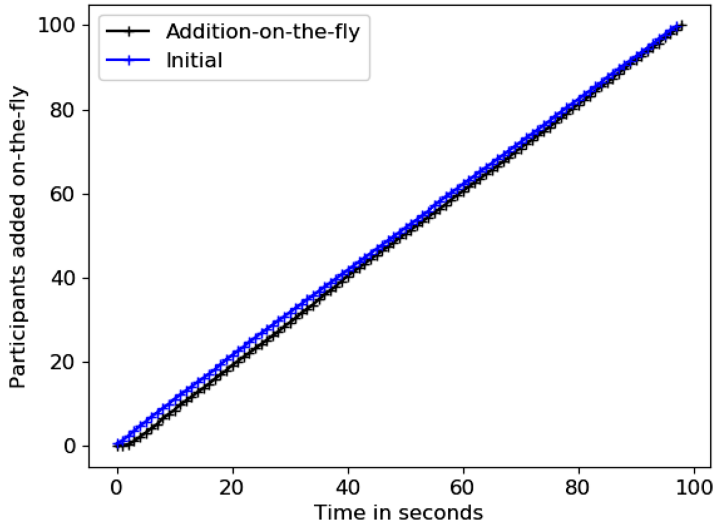

Fig. 7. Time needed to complete a given task using on the fly addition of nodes.

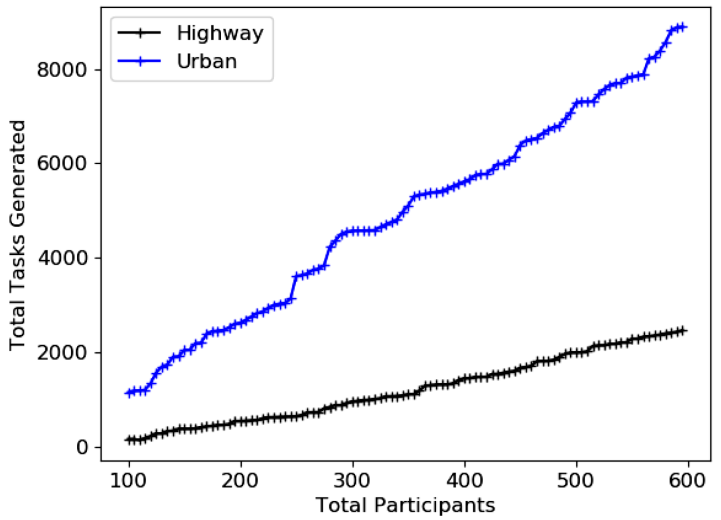

Fig. 8. Total number of tasks generated.

6.1. Results & discussion

Some valuable insights were observed through these simulations. Although the maximum density was around 1000 nodes in urban areas, whereas in highways it is comparatively low, the nodes willing to participate were a subset of that. Participation varies between scenarios. In the case of low density of participants, the algorithm performance shows certain fluctuations as explained below. Following a greedy allocation always assists in a quick search, if there are no possible assignments. Further, varying the arrivals tends to increase reward function in essence showing a high profit margin. The system always tries to satisfy more requests, inspite of the initial systems allocation of resources in the cloud. Therefore, the system adjusts service requests demands. Variation of the arrival for different values shows this pattern as observed in Fig. 6.

Platooning is a vehicular terminology that makes one vehicle control and lead the speed of the vehicles behind it. In Fig. 11, we observe that higher the velocity of the platooning leader, the lower the bandwidth utilization. While calculating the requests off-loaded, we use the following formula

$$Requests - Offloaded = \frac{No.\ of\ accepted\ requests}{Total\ requests\ generated} \tag{10}$$

We observe an average of up-to 67% successful offloads with our framework owing to the knowledge of the resource DB. However, this value is around 52% for the highway scenario but still performs better than local processing. Details are depicted in Fig. 12.

Profit margin for the RSU operator depends on the contribution of the nodes. Fig. 13 shows a margin of over 70% profit is

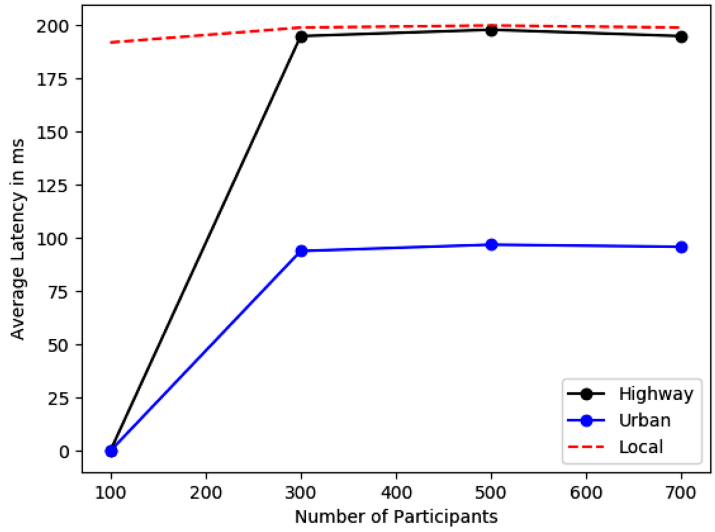

Fig. 9. Average Latency Observed.

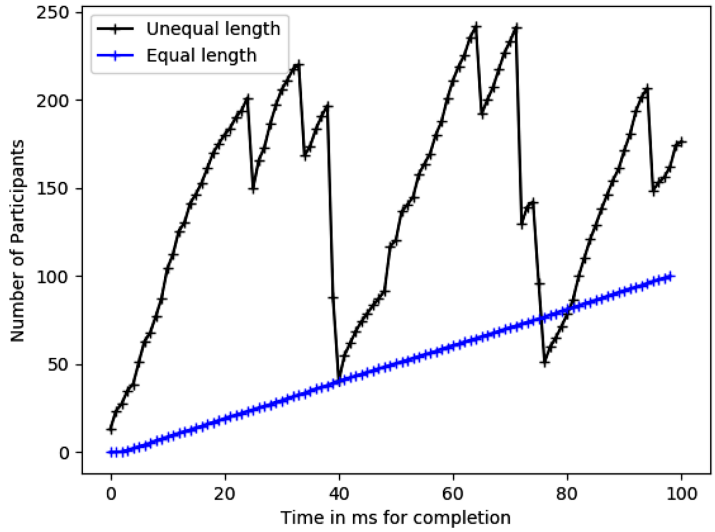

Fig. 10. Job completion time for Equal length and Unequal length jobs with varying Node participation.

observed in urban scenarios. A lower margin is observed in highway scenarios which can be attributed to low node density. The entire cost modelling is done by taking into consideration that at least 10% of resources are share-able by all nodes. A minimum profit value is measured per-contribution and is set to a default value of 25c/$/h. The margin% calculated is determined according to the following:

$$Margin\% = (Profit - Cost)/Profit \tag{11}$$

The cost price however varies based on the location of the RSU and is based the previous successful job attempts. For this work, we do not consider per node profit distribution. Our aim is to model the total profit gained by the RSU in order for the RSU to be part of this cloud 'brokering'-like scenario. Our evaluations shows that RSU operators indeed profit from this method. We can observe that decrease in reward is attributed to vehicles leaving. Most certainly vehicular cloud migration comes into play in such circumstances. Thus, the system is affected adversely. However, we deal with this limitation by maintaining the RSU link that makes another vehicle available immediately. In this way we try to maintain the gain.

6.2. Comparison against state-of-the-art solutions

We compared our proposed design with a state-of-the-art Availability-Based Resource Allocation Scheme for Vehicular Cloud (AVARAC) [20]. Here, the reward is determined as the lump sum income gained by the system. Likewise, we map it to the profit

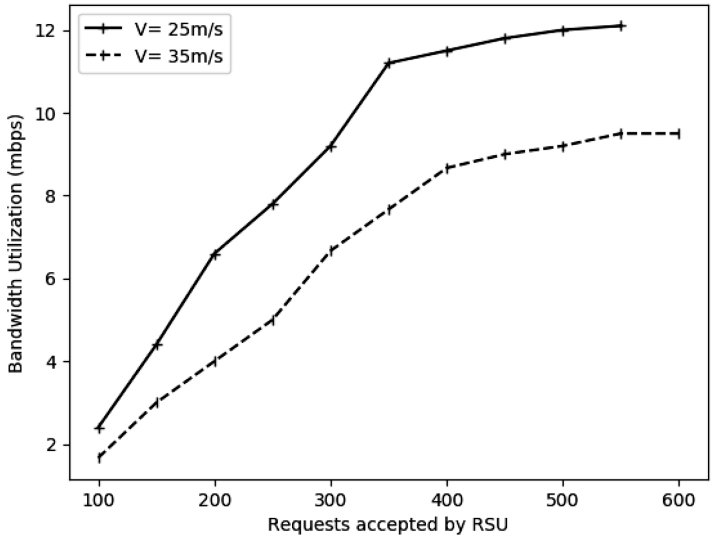

Fig. 11. Effect of platooning on Bandwidth.

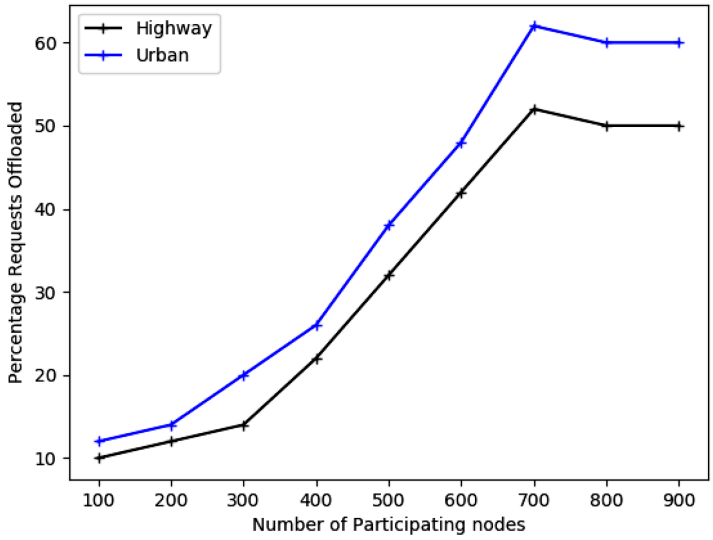

Fig. 12. Percentage of requests offloaded.

margin gained by the operator in the event of such a subscription. In order to create such a mapping, we model the vehicle leaving rate $lv_{\{veh\}}$, that provides the information of resources withdrawn from the cloud. Furthermore, we keep resource requisitions similar to Meneguette et al. [20] to maintain similarity in terms of resource allocation parameters. As there are no available implementations for a one-to-one comparison, we model these parameters in our simulation.

For our evaluation, we use two resource class quantities namely $b_1 = 2, 3$ *and* $b_2 = 2, 3$. In our work, we statically define these quantities as it is not clear in [20] how the choice of class is made. Furthermore, our evaluation strictly takes into consideration the urban scenario for comparison only, as is the case in [20].

Our proposed Low-Latency Vehicular Infrastructure Framework (LL-VIF) performs better owing to the design of the mobility tracker that engages in keeping the resource database updated. Due to this, the vehicles leaving the region do not always negatively affect the profit of the system. However, in AVARAC, the system is modelled based on static resource allocation, hence, a resource that leaves will not have a replacement, which leads to a drop in profit. As observed in Fig. 14 We notice over 10% increase in profit margin compared to AVARAC.

From Fig. 15, we see that the leaving rate of vehicles does not impact our framework as much as it does using the AVARAC model. This can be attributed to the presence of a constantly updated congestion manager that creates local buffers for tasks that need to be further re-processed owing to the leaving rate.

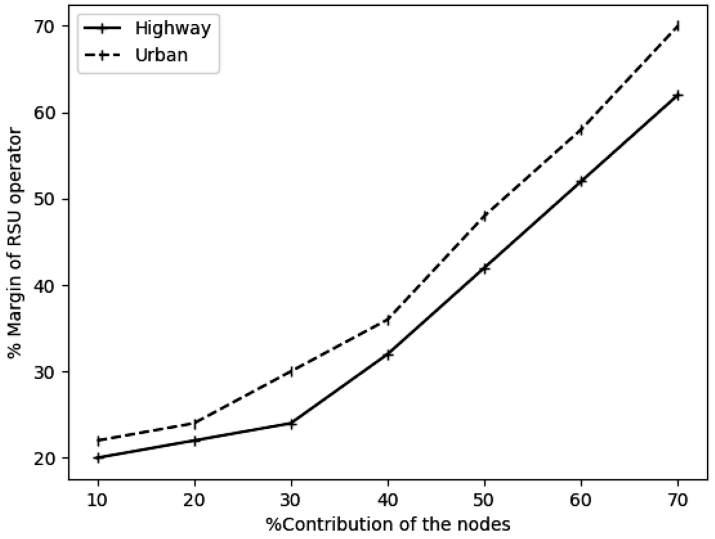

Fig. 13. Profit margin.

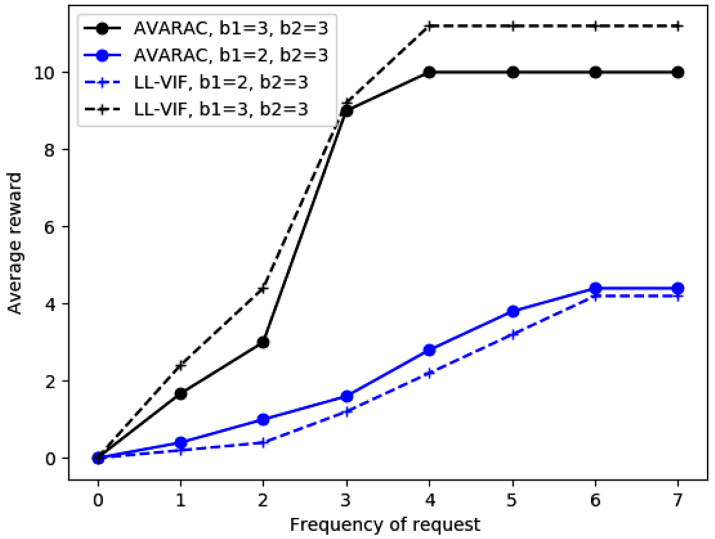

Fig. 14. Reward variation with Frequency of Requests.

7. Conclusion and future work

In this paper, we proposed an Road Side Unit (RSU)-based edge solution for on-the-fly vehicular infrastructure provisioning. An architecture is envisioned for managing the vehicular infrastructure without having any central controller as seen in software defined environments. Every RSU plays the role of a broker that performs job assignments. The proposed framework is compared with two competing solution where the job assignments are done locally, in highways and in urban areas. Our framework shows considerable improvements in terms of latency measurements and bandwidth utilization. We bring in a novel approach of adding new nodes on-the-fly as seen in typical virtual embedding scenarios.

This framework will be extended in the future to Network Slicing paradigms. As we are heading towards the future of 5G and beyond, service guarantees at sub-microsecond latency is inevitable. By utilizing idle device resources for formation of “edge cloud” of vehicles, we take a step forward in making “on-the-fly” computing a reality, where service guarantees are typically going to be one-hop away. In the future, we also plan to investigate decision making processes within the queuing time for offloading between multiple composition services. The RSU’s collaborating between each other may sometimes be inter-connected which can be further exploited for scaling up the composition service guarantees. Scheduling in such scenarios become difficult according to job characteristics and network connections, and hence is another potential direction for future research.

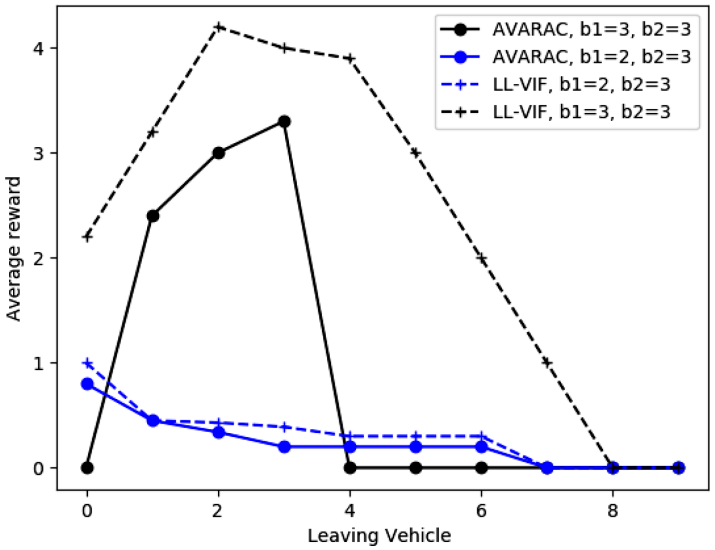

Fig. 15. Reward variation Leaving Rate.

References

[1] A.A. Alkheir, M. Aloqaily, H.T. Mouftah, Connected and autonomous electric vehicles (caevs), IT Prof. 20 (6) (2018) 54–61.
[2] I.A. Ridhawi, M. Aloqaily, B. Kantarci, Y. Jararweh, H.T. Mouftah, A continuous diversified vehicular cloud service availability framework for smart cities, Comput. Netw. 145 (2018) 207–218, https://doi.org/10.1016/j.comnet.2018.08.023.
[3] Number of motor vehicles registered(https://www.statista.com/statistics/183505/number-of-vehicles-in-the-united-states-since-1990/). 2018.
[4] Ave stats(https://www.statista.com/topics/3573/autonomous-vehicle-technology/). 2018.
[5] S. Toglaw, M. Aloqaily, A.A. Alkheir, Connected, autonomous and electric vehicles: the optimum value for a successful business model, 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, (2018), pp. 303–308, https://doi.org/10.1109/IoTSMS.2018.8554391.
[6] Cruise-av-drive without steering, (https://www.cnet.com/roadshow/news/cruise-av-no-steering-wheel/), 2018.
[7] G.A. McGilvary, A. Barker, M. Atkinson, Ad hoc cloud computing, 2015 IEEE 8th International Conference on Cloud Computing, (2015), pp. 1063–1068, https://doi.org/10.1109/CLOUD.2015.153.
[8] A. Mtibaa, K.A. Harras, A. Fahim, Towards computational offloading in mobile device clouds, 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, 1 (2013), pp. 331–338, https://doi.org/10.1109/CloudCom.2013.50.
[9] V. Balasubramanian, A. Karmouch, An infrastructure as a service for mobile ad-hoc cloud, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), (2017), pp. 1–7, https://doi.org/10.1109/CCWC.2017.7868393.
[10] J. Feng, Z. Liu, C. Wu, Y. Ji, Ave: autonomous vehicular edge computing framework with aco-based scheduling, IEEE Trans. Veh. Technol. 66 (12) (2017) 10660–10675.
[11] N. Lu, N. Cheng, N. Zhang, X. Shen, J.W. Mark, Connected vehicles: solutions and challenges, IEEE Internet Things Journal 1 (4) (2014) 289–299.
[12] M. Aloqaily, V. Balasubramanian, F. Zaman, I. Al Ridhawi, Y. Jararweh, Congestion mitigation in densely crowded environments for augmenting qos in vehicular clouds, Proceedings of the 8th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, DIVANet'18, ACM, New York, NY, USA, 2018, pp. 49–56, https://doi.org/10.1145/3272036.3272038.
[13] V. Balasubramanian, N. Kouvelas, K. Chandra, R. Prasad, A. Voyiatzis, W. Liu, C. Xu, X. Deng, L. Zhang, J. Fang, et al., A unified architecture for integrating energy harvesting iot devices with the mobile edge cloud, IEEE 4th World Forum on Internet of Things (WF-IoT), 56 IEEE, 2016, pp. 1469–1499.
[14] Talk to me: the present & future of in-car speech recognition, (https://www.globalme.net/blog/the-present-and-future-of-in-car-speech-recognition/). 2018.
[15] E. Massaro, et al., The Car as an Ambient Sensing Platform [Point of View], Proceedings of the IEEE, 105 (1) (2017), pp. 3–7.
[16] I.A. Ridhawi, N. Mostafa, Y. Kotb, M. Aloqaily, I. Abualhaol, Data caching and selection in 5g networks using f2f communication, 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), (2017), pp. 1–6, https://doi.org/10.1109/PIMRC.2017.8292681.
[17] I. Al Ridhawi, M. Aloqaily, Y. Kotb, Y. Al Ridhawi, Y. Jararweh, A collaborative mobile edge computing and user solution for service composition in 5g systems, Trans. Emerg. Telecommun.Technol. 29 (11) (2018) e3446.
[18] Y. Zhiwen, Z. Xingshe, Z. Daqing, An adaptive in-vehicle multimedia recommender for group users, Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st, 5 IEEE, 2005, pp. 2800–2804.
[19] S. Rene, E. Exposito, M. Gineste, J. Alins, O. Esparza, Multipath tcp architecture for infotainment multimedia applications in vehicular networks, Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st, IEEE, 2015, pp. 1–5.
[20] R.I. Meneguette, A. Boukerche, A.H. Pimenta, Avarac: an availability-based resource allocation scheme for vehicular cloud, IEEE Trans. Intell. Transp.Syst. (2018).
[21] Y. Kotb, I. Al Ridhawi, M. Aloqaily, T. Baker, Y. Jararweh, H. Tawfik, Cloud-based multi-agent cooperation for iot devices using workflow-nets, J. Grid Comput. (2019), https://doi.org/10.1007/s10723-019-09485-z.
[22] I. Al Ridhawi, M. Aloqaily, Y. Kotb, Y. Jararweh, T. Baker, A profitable and energy-efficient cooperative fog solution for iot services, IEEE Trans. Ind. Inf. (2019), https://doi.org/10.1109/TII.2019.2922699. 1–1
[23] Y. Mao, J. Zhang, K.B. Letaief, Dynamic computation offloading for mobile-Edge computing with energy harvesting devices, IEEE J. Sel. Areas Commun. 34 (12) (2016) 3590–3605, https://doi.org/10.1109/JSAC.2016.2611964.
[24] M. Satyanarayanan, P. Bahl, R. Carceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Comput. 8 (4) (2009) 14–23, https://doi.org/10.1109/MPRV.2009.82.
[25] B. Venkatraman, F.A. Zaman, A. Karmouch, Optimization of device selection in a mobile ad-hoc cloud based on composition score, 2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA), (2017), pp. 257–262, https://doi.org/10.1109/CSCITA.2017.8066564.
[26] C. Paasch, G. Detal, F. Duchene, C. Raiciu, O. Bonaventure, Exploring mobile/wifi handover with multipath tcp, Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design, CellNet '12, ACM, New York, NY, USA, 2012, pp. 31–36, https://doi.org/10.1145/2342468.2342476.

[27] D. Zhu, C. Xu, J. Qin, Z. Zhou, J. Guan, Mobility-aware multimedia data transfer using multipath tcp in vehicular network, Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International, IEEE, 2017, pp. 1041–1046.
[28] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, I. Khalil, Datacenter at the airport: reasoning about time-dependent parking lot occupancy, IEEE Trans. Parallel Distrib.Syst. 23 (11) (2012) 2067–2080.
[29] K. Wang, H. Yin, W. Quan, G. Min, Enabling collaborative edge computing for software defined vehicular networks, IEEE Netw. (2018).
[30] W. Zhao, J. Liu, T. Hara, Optimal replica distribution in edge-node-assisted cloud-p2p platforms for real-time streaming, IEEE Trans. Veh. Technol. (2018).
[31] J. Feng, Z. Liu, C. Wu, Y. Ji, Hvc: a hybrid cloud computing framework in vehicular environments, 2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), (2017), pp. 9–16.
[32] H. Choi, Y. Nam, J. Bang, E. Lee, Multi-hop vehicular cloud construction with connection time based resource allocation in vanets, 2018 24th Asia-Pacific Conference on Communications (APCC), (2018), pp. 100–105.
[33] Y. Yao, X. Chang, J. Mišić, V. Mišić, Reliable and secure vehicular fog service provision, IEEE Internet Things J. 6 (1) (2019) 734–743.
[34] Z. Su, Y. Hui, S. Guo, D2d-based content delivery with parked vehicles in vehicular social networks, IEEE Wireless Commun. 23 (4) (2016) 90–95.
[35] M. Steiner, T. En-Najjary, E.W. Biersack, Exploiting kad: possible uses and misuses, ACM SIGCOMM Comput. Commun. Rev. 37 (5) (2007) 65–70.
[36] M. Popovici, C. Raiciu, Exploiting multipath congestion control for fun and profit, Proceedings of the 15th ACM Workshop on Hot Topics in Networks, HotNets '16, ACM, New York, NY, USA, 2016, pp. 141–147, https://doi.org/10.1145/3005745.3005769.
[37] M. Gerla, J.-T. Weng, G. Pau, Pics-on-wheels: photo surveillance in the vehicular cloud, Computing, Networking and Communications (ICNC), 2013 International Conference on, IEEE, 2013, pp. 1123–1127.
[38] S.M. Ross, Introduction to Stochastic Dynamic Programming, Academic Press, 2014.
[39] C. Sommer, R. German, F. Dressler, Bidirectionally coupled network and road traffic simulation for improved IVC analysis, IEEE Trans. Mobile Comput. 10 (1) (2011) 3–15, https://doi.org/10.1109/TMC.2010.133.
[40] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of sumo-simulation of urban mobility, Int. J. Adv. Syst.Meas. 5 (3&4) (2012).
[41] M.L. Sichitiu, M. Kihl, Inter-vehicle communication systems: a survey, IEEE Commun. Surv. Tutorials 10 (2) (2008).