The trade-offs between Fog Processing and Communications in latency-sensitive Vehicular Fog Computing

Francisco Mendonça Júnior

Zafeiris Kokkinogenis

Kelvin Dias

Pedro d'Orey*

Rosaldo J. F. Rossetti

*CISTER Research Centre

CISTER-TR-220607

2022/06/16

The trade-offs between Fog Processing and Communications in latency-sensitive Vehicular Fog Computing

Francisco Mendonça Júnior, Zafeiris Kokkinogenis, Kelvin Dias, Pedro d'Orey*, Rosaldo J. F. Rossetti

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: ffmj@cin.ufpe.br, kld@cin.ufpe.br, ore@isep.ipp.pt

https://www.cister-labs.pt

Abstract

In a vehicular fog computing paradigm, connected autonomous vehicles are envisioned as processing nodes (i.e. fog nodes) so that end-devices may offload processing tasks to them. As such, both local and distributed processing on fog nodes will depend heavily on wireless network conditions and the current traffic demand. In this work, we investigate the trade-offs on the operation of fog nodes under different vehicle densities and network conditions and formalize a Time Constrained One-Shot Open First Price Auction for resource allocation in vehicular fog computing. Through a large-scale simulation study, we assess important aspects of the performance of fog nodes in Vehicular Fog Computing. We show that current wireless network standards may dictate the limits of processing despite the availability of processing power of fog nodes. Our results indicate the existence of trade-offs on the operation of fog nodes regarding message overhead and processing redundancy to achieve high task completion ratio. Finally, we evaluate the social welfare distribution of the task allocation achieved using the auction where higher message rates lead to higher costs.

Pervasive and Mobile Computing 84 (2022) 101638

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

The trade-offs between Fog Processing and Communications in latency-sensitive Vehicular Fog Computing

Francisco Ferreira de Mendonça Junior [a,*], Zafeiris Kokkinogenis [b,c], Kelvin Lopes Dias [a,*], Pedro M. d'Orey [d,e], Rosaldo J.F. Rossetti [b,c]

[a] Center of Informatics, Federal University of Pernambuco, Recife, Brazil
[b] Artificial Intelligence and Computer Science Lab, University of Porto, Portugal
[c] Faculty of Engineering, University of Porto, Porto, Portugal
[d] CISTER Research Center, Porto, Portugal
[e] Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto, Rua do Campo Alegre s/n, 4169-007 Porto, Portugal

ARTICLE INFO

Article history:
Received 21 September 2021
Received in revised form 10 May 2022
Accepted 10 June 2022
Available online 16 June 2022

Keywords:
Vehicular fog computing
Fog computing
Heterogeneity
Simulation
Auctions

ABSTRACT

In a vehicular fog computing paradigm, connected autonomous vehicles are envisioned as processing nodes (i.e. fog nodes) so that end-devices may offload processing tasks to them. As such, both local and distributed processing on fog nodes will depend heavily on wireless network conditions and the current traffic demand. In this work, we investigate the trade-offs on the operation of fog nodes under different vehicle densities and network conditions and formalize a Time Constrained One-Shot Open First Price Auction for resource allocation in vehicular fog computing. Through a large-scale simulation study, we assess important aspects of the performance of fog nodes in Vehicular Fog Computing. We show that current wireless network standards may dictate the limits of processing despite the availability of processing power of fog nodes. Our results indicate the existence of trade-offs on the operation of fog nodes regarding message overhead and processing redundancy to achieve high task completion ratio. Finally, we evaluate the social welfare distribution of the task allocation achieved using the auction where higher message rates lead to higher costs.

1. Introduction

Internet of Vehicles (IoV) [1] has emerged as the next step of the evolution of both Vehicular Ad Hoc Networks (VANETs) and Mobile Networks as it encompasses Intelligent Transportation Systems (ITS) and applications for users, i.e. passengers. Also, Vehicular Fog Computing (VFC) [1] comes up both as a cause to and a consequence of IoV, as vehicles may provide processing capabilities closer to end devices, namely other vehicles, for the passenger, and/or surrounding users.

With the emergence of media stations with high processing power inside vehicles, enhanced connectivity, and even autonomous vehicles, VANETs may support an entirely new set of objectives and applications such as advertisement and infotainment, as well as enhanced driving functionalities. These applications combined with autonomous driving application demand intense exchange of large messages containing images or video frames while simultaneously imposing strict delay requirements for various tasks (e.g. video crowd-sourcing, traffic safety [2–4]).

* Corresponding authors.
E-mail addresses: ffmj@cin.ufpe.br (F.F. de Mendonça Junior), zafeiris.kokkinogenis@gmail.com (Z. Kokkinogenis), kld@cin.ufpe.br (K.L. Dias), ore@isep.ipp.pt (P.M. d'Orey), rossetti@fe.up.pt (R.J.F. Rossetti).

https://doi.org/10.1016/j.pmcj.2022.101638
1574-1192/© 2022 Elsevier B.V. All rights reserved.

VFC enables vehicles to provide information and serve each other as processing units within the network. In this context, the basic operation of a VFC consists of vehicle servers (i.e. fog nodes) that respond to client requests from mobile users, Internet of Things (IoT) devices, or other vehicles, which have been received via wireless communications. We argue that Edge and Fog Servers' operation in VFC is especially suitable for time-critical applications and may depend on the lightest available negotiation protocols [5]. Recent studies pointed to the increasing complexity [6,7] in managing such networks, especially due to their high and intricate mobility of nodes. We argue that the coexistence of diverse applications running on heterogeneous devices requires an in-depth analysis of the devices' operation and the network itself.

This paper extends our previous work [8] by presenting a study on how vehicles operate while acting as servers in VFC. Also, through a systematic review, an overview of the related resource management literature is given. Specifically, we investigate the impacts and limits of such operation regarding the processing capacity of single and clustered fog nodes, and how the network capacity may enable or hinder the adoption of groups of applications. The main paper contributions can be summarized as follows:

- in-depth study of the operation of fog nodes under diverse network conditions and application requirements in a completely distributed VFC;
- formalization and analysis of the suitability of a Time Constrained One-Shot Open First Price Auction (TC-OSOFPA) to coordinate the allocation for processing of latency-sensitive applications;
- comparison between the TC-OSOFPA and a piggyback method with respect to the (a) overheads each approach introduces in the task allocation and (b) social welfare distribution of the allocation;
- discussion and evaluation of the limitations the network imposes on a fully distributed VFC.

The remainder of this paper is organized as follows. Section 2 presents the relevant related work in the field, being followed by the discussion of the system requirements, assumptions, and proposals in Section 3. Section 4 outlines the evaluation strategy and discusses the results. Finally, Section 5 concludes this paper and outlines plans for future work.

2. Related work

Two high correlated perspectives encompass the operation of VFC: resource allocation and task assignment. Both perspectives may require anticipating network behavior, managing and virtualizing computational and radio resources, thus increasing computational complexity [6]. Bearing that in mind, we select works based on a systematic review with some inclusion criteria: works focused on load balancing in VFC in which vehicles are active processing devices [19] (see Table 1). Selected works focus on resource management in VFC investigate how to reduce latency [9–11,13], improve resource availability [15,16,20], and validate methods [5].

Still, none of them, as they were designed, are suitable for latency-sensitive applications according to ITS requirements as we can get from the lack of evaluation at network level and demonstration of latency constraints [5,7,14–17,20] or latency constraints above 100 ms [9–12,18]. We argue that latency is actually one of the main benefits of using VFC, alongside the decentralization, collaboration, and the independence from the Cloud.

Most of them also leverage third parties, such as Road Side Unit (RSU), Base Station (BS) or even the Cloud, to manage resource allocation and task assignment; only three of such works presents a completely distributed architecture [17,18, 20]. In this paper, we addressed the open issues of VFC especially following the definitions of Fog [21] and IoV [1] which claim that both of them should be heterogeneous, wireless, autonomous, omnipresent, decentralized, and cooperative.

Authors in [9] investigate how to minimize the latency on a Fog-enabled application for real-time traffic management by balancing the load amongst Cloud, cloudlets, Vehicular Cloud Computing (VCC), and VFC leveraging a series of algorithms such as branch-and-bound and Edmonds–Karp. They reduce response time when compared to a random load balancing and task distribution approach. Although presenting the negotiation protocols, they do not consider network behavior such as delivery failures and packet loss in crowded environments.

Xu et al. investigate in [10] how to reduce service latency while investigating the task assignment problem by proposing a price-based matching algorithm. The authors do not consider the impacts of network operation on negotiations to build a list of preferred servers before the actual service starts. The results indicate that service latency increases with the number of User Equipment (UE) as they need more iterations to reach an optimal solution. On the other hand, delay decreases with an increase in the number of iterations under a fixed number of clients. Under those conditions, we argue that such a system seems unfeasible to latency-sensitive applications as negotiations may delay the actual delivery of responses. The study does not consider the impacts of network operation on negotiations as it builds a list of preferred servers before the actual service starts.

Zhou et al. [11] leverage a contract-theory and price-based stable matching algorithm to reduce service latency. Delay increases with the number of types of contracts, the number of UEs, and with the speed of Fog Nodes. On the other hand, delay decreases with an increase in the number of iterations and with the loosening of the delay constraints. Thus, the proposed negotiation mechanism is not suitable for latency-sensitive applications and, once again, users may not receive their responses in time.

The works in [12,13] discuss how Folo (Fog following me) aims to reduce latency and loss of quality by dynamically allocating event-triggered tasks. The authors argue that VFC presents a trade-off between service latency and quality of

Table 1
Comparison of related work.

Reference	Problem	Proposal	Fog Only?	Delay Req.
Ning et al. 2019 [9]	VFC for traffic management towards reduced communication latency	Branch and bound algorithm for load balancing amongst Cloud, Cloudlet, VCC e VFC	NO	200∼500 ms
Xu et al. 2018 [10]	Task assignment challenges aiming to reduce latency	Pricing-based Matching theory	NO	>100 ms
Zhou et al. 2019 [11]	Incentive in task assignment to reduce service latency	Pricing-based stable matching algorithm	NO	>100 ms
Zhu et al. 2018 [12,13]	Trade-off between latency and quality	Event-triggered task allocation problem	NO	0∼1000 ms
Lin et al. 2018 [14]	Service Latency reduction by increasing overall system utility	Utility model and two-step optimization	NO	NA
Klaimi et al. 2018 admission	Potential theoretical game, scheduling algorithm, decentralized game decision	YES	NA	
Yao et al. 2018 [15]	Random arrivals and departures analysis in VFC to increase reliability and security	VFC construction method and access method leveraging TA	NO	NA
Zhang et al. 2019 [16]	Increase Resource Availability	Auction for reservation of parking places to Fog Nodes	NO	NA
Zhao et al. 2019 [7]	Increase vehicle participation in VFC for burden reduction from RSU (Edge)	Deep reinforcement learning to decide about task assignment	NO	NA
Peng et al. 2020 [5]	Lack of investigation about auction in VFC to increase computational efficiency, individual rationality, budget balance, and truthfulness.	Multiattribute-based double auction	NO	NA
Kang et al. 2020 [17]	Task prioritization in VFC	Game theory and Lagrange dual method	YES	NA
Cha et al. 2021 [18]	Scarcity of computational resources	Clustering	YES	NA

the response which emerges as an optimization problem. They investigate how clients, servers, and infrastructure (Edge, RSU) may negotiate to enable efficient service discovery and server selection for video streaming and real-time object detection applications. The results indicate that linear programming is better in reducing latency, whereas swarm particle optimization keeps the quality levels high. Nevertheless, they still keep delay constraints above 100 ms, while our paper investigates scenarios and mechanisms that can deliver results below 50 ms.

Lin et al. [14] aim to reduce service latency by allocating network bandwidth to four different types of services: Traditional Elastic Services (data transmission, HTTP), Interactive Elastic Services (online chatting), Hard real-time services (VoIP), and Soft real-time services (Live Streaming). The authors propose a two-step solution to allocate those resources. However, they do not account for negotiation among fog nodes to allocate the tasks, thus leaving room for this investigation.

Although not investigating latency issues in VFC, [20] aims at minimizing packet loss and increase task admission rate for high priority applications under dynamic vehicle resources. They use a game-theoretic approach to solve the problem of arrival and departures of vehicles in parking lots. Although they claim to have reduced the latency, the authors do not present numerical evaluation of reduction, nor the delay constraints of applications involved.

In [15], authors are concerned with the randomness of arrivals and departures of parked vehicles. They propose a VFC framework based on containers with both a construction method and an access method based on the negotiations with a Trusted Authority (TA). Although secure, such a framework is not well suited for latency-sensitive applications as it may take too long to access the resources of VFC.

The study in [16] aims to increase resource availability in VFC through an auction-based parking mechanism. It works as an incentive for Fog Nodes to park in specific locations and provide their services to the clients in the vicinity. Fog Nodes receive park location, and also receive incentives for lending processing power. The available resources increase as the offloading price increases. On the other hand, negotiations require more rounds as the number of servers requiring parking increases. Also, the paper does not present the evaluation of latency on clients nor servers.

Zhao et al. consider in [7] the lack of available resources and incentives in VFC, as well as the complexity of task offloading and possible offloading collision. The Fog Nodes use Deep Reinforcement Learning to decide whether to respond to requests after receiving a reward contract from the RSU. The normalized system performance is based on system delay, energy consumption, task completion rate, and vehicle contribution resources. Thus, we cannot actually evaluate delay constraints of such a solution nor assess the computational complexity of the framework.

Kang et al. [17] argue that VFC natively has services that require prioritization. On their model, tasks are dynamically priced and prioritized according to the application's requirements, features, and network conditions. The authors address the allocation problem leveraging game theory and the dual Lagrange method, always considering mobility and task deadline. The solution presents low utilization rate, high completion rate, and small average pricing when compared to three other solutions, i.e., LPS (Local Processing Scheme). However, unlike our work, the success rate of responses increases with server density. We argue that this results from the lack of evaluation at the network level.

Cha et al. [18] evaluate a Virtual Edge, i.e., a cluster of server vehicles that provide its processing in a distributed way. In the evaluation, this solution is compared with random models, with minimum distance and maximum resource, having presented better results in the maintenance of the clusters in response latency as the number of servers increases. However, the solution still evaluates applications with relaxed latency requirements above 5s.

Finally, Peng et al. [5] investigate resource allocation in VFC relying on a multiattribute-based double auction approach. They primarily evaluate the performance of the mechanism based on location, reputation, and computing power of Fog Nodes to set client–server pairs. However, the authors do not consider all dimensions in their network performance evaluation. The obtained results might not be feasible nor reflect other scenarios under some vehicular networks' constraints, such as message collision and bit error rates. Thus, we argue that further evaluations might be needed under different network conditions.

3. On the operation of delay sensitive applications in VFC

3.1. Assumptions on operation

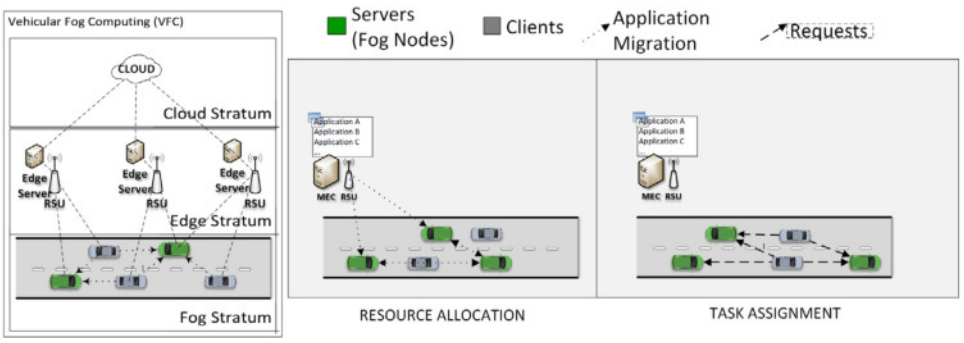


Fig. 1. Vehicular fog computing strata and resource allocation and task assignment.

Fig. 1 displays the basic operation and strata of VFC according to FogWise model [8]. In the VFC paradigm three processing stratum are considered: (i) *fog stratum* wherein (selected) vehicles act as processing nodes,(ii) *edge stratum* wherein roadside infrastructure (e.g. RSU) with communication and processing capabilities conduct tasks, and (iii) *cloud stratum*. In this work, we focus on the *fog stratum* leaving the interactions or cooperation with other strata for future work. Initially, sensors, mobile users or other vehicles (i.e. Clients) might need to schedule (cloud) processing services, such as Platform as a Service (PaaS), Infrastructure as a Service (IaaS), or Software as a Service (SaaS). Some of those endpoints may wish to save resources such as computing, energy, or networking. In this context, the *resource allocation problem* comprises the mechanisms that servers have available to download the applications and other resources that clients may need. On the other hand, users need a way to discover services and send requests to servers in the vicinity that will negotiate in advance or in real-time to respond to requests: *the task assignment problem*, i.e. users select a server from a pool of processing resources based on a number of criteria (e.g. response time, cost). To enable short response times a mechanism to download in advance the required services is also required.

The following set of assumptions supported by the current state-of-the-art are considered:

1. resource allocation problems may present minor impact, as the number of apps may be limited as few apps stand as popular[1];
2. fog nodes carry metadata about applications in the form of a profiling database that can be queried to estimate the execution time of applications [22];
3. the best suited model for Fog Computing is the Request - Processing - Response – Actuation (RPRA) model [23] where servers receive a request and return a response as fast as possible;
4. data exchanged amongst vehicles incorporate security mechanisms;
5. energy consumption is not a hindrance for moving VFC as vehicles may acquire energy from several sources [24]. Still, there is room to optimize energy consumption and distribution [7,20];
6. collisions and network errors (RXTX and SNIR) increase (exponentially) with network density, request rate and request size [8], which may be usual during e.g. traffic jams.

3.2. The servers

We focus on the operation of fog nodes, i.e., servers under latency-sensitive applications. Algorithm 1 presents the simplified operation of fog nodes. Server S_i is constantly listing for incoming message and at a given time receives a message m bearing a clients' application identification, i.e *uuid* code (line 2). The messages can be of several types: (i) *task processing requests* sent by clients, (ii) *bids* sent by either other servers or auctioneers, and (iii) *responses* sent by servers containing a processed task. If the message is of type *task processing request* the server adds the request into a queue for further processing, and broadcasts its bid (i.e S_i.bid[uuid]) containing the required processing time (lines 3–6). The processing time bid is an estimation based on server's profiling database. The more requests it receives, the more queuing time increases, possibly delaying the following responses. If the message is of type *bid* sent by an auctioneer with the adjudication of the task *uuid* to server S_i (line 7), then S_i starts the task processing (lines 8–9), or otherwise removes it from its queue (line 10–12). In an asynchronous fashion, the server S_i processes its assigned (active) tasks. When the processing is completed the server broadcasts a response and removes the task from its queue (lines 19–22).

Algorithm 1: Simplified server operation

```
   Input: server $S_i \in S$,list of messages $M$, $m \in M$,list of bids,list of active tasks
   Output: Processing task assignment based on winning bids
 1 while $S_i$ listens do
       /* Thread 1                                                    */
 2     foreach message $m$ do
 3         if $m == request$ then
 4             enqueueForProcessing(m[uuid])
 5             sendBid()
 6         end
 7         else if $m == bid$ then
 8             if $Winner.bid[uuid] == S_i.bid[uuid]$ then
 9                 startProcessing(m[uuid])
10             else
11                 remove(m[uuid]);
12             end
13         end
           // message is a response
14         else
15             remove(m[uuid]);
16         end
17     end
18 end
   /* Thread 2                                                        */
19 while $S_i$ has active processing tasks do
20     if m[uuid].finished then
21         sendResponse(r[uuid])
22         remove(m[uuid]);
23     end
24 end
```

Algorithm 2 presents the client side of the VFC operation. Combined, both algorithms leverage the widely known Fire and Forget [25] pattern as the basis for operation. Thus, we associated these communication and processing characteristics to build a *fully distributed* auction, where clients are also auctioneers and contain the internal logic to control the auction. The operation of client C_i involves three asynchronous execution tasks that run in parallel. In task 1, the client broadcasts requests for processing transmitting message $m[uuid]$ (lines 1–3) to benefit from the one-hop principle of vehicular networks [13] and thus avoid routing and relaying. Servers in the vicinity receive the requests and estimate response time based on a profiling database. In task 2, a client C_i that has requested the processing of a task *uuid* receives the

[1] www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-2017-US-Mobile-App-Report, Accessed January 8, 2022.

related bids and determines the winner of the auction later broadcasting the winning bid (lines 4–6). In task 3, C_i receives the server responses regarding the completion of the requested task (lines 7–13).

Algorithm 2: Simplified client operation

```
Input: client $C_i \in C$, server $S_j \in S$ list of Server's responses $r \in R$, Client message $m$
Output: Client's application processing
/* Thread 1                                                    */
1  foreach Client's $C_i$ application uuid running do
2      $C_i$.sendRequest(m[uuid])
3  end
/* Thread 2                                                    */
4  while Client $C_i$ has open auctions do
5      $Winner.bid[uuid] \leftarrow runAuction(uuid)$
6  end
/* Thread 3                                                    */
7  foreach response $r$ do
8      if $S_j.r[uuid] == C_i.m[uuid]$ then
9          accept($S_j.r[uuid]$)
10     else
11         ignore($S_j.r[uuid]$)
12     end
13 end
```

3.3. Auction model and problem formulation

Combined, Algorithms 1 and 2 constitute a Time Constrained One-Shot Open First Price Auction (TC-OSOFPA) [8] that starts after clients $C = \{C_1, C_2, C_3, \ldots, C_n\}$ broadcast requests containing an Ask for processing (A_n). This Ask contains at least the load that applications may inflict over the processing power of servers. Servers $S = \{S_1, S_2, S_3, \ldots, S_m\}$ estimate processing time based on the load from A_n and broadcast a bid containing the estimated time. The network message also contains a request-id and the current position that may be used in future implementations. Let, H_m be the bid of the S_m server. Eq. (1) demonstrates that the response time from a bid is determined by dividing the load of the application from (A_n) and the processing power (PP_m) of the vehicle added by the sum of the previous tasks it is currently processing, that is when there is a processing queue on the Fog Node. This is an open bid, henceforth other servers may cancel the bid or the task if their internal estimate is higher than the bid they perceive from the network.

$$H_m = \frac{A_n}{PP_m} + \sum_{0}^{n-1} \frac{A_n}{PP_m} \tag{1}$$

In this model, the winner is determined according to Eq. (2). The winner (W_n) in the auction is the server that sends the bid containing the smallest estimate response time within time constraints. Thus, if at least one timely bid $T(H_m)$ exists within the requirements of the client $R(n)$, it is considered the best bid. A tiebreaker rule mandates that the client chooses randomly when there are repeated winning bids.

$$W_n = \underset{m}{\operatorname{argmin}}(H_m), \exists H_m : T(H_m) < R(n) \tag{2}$$

This auction is *time constrained* as clients may receive a response even if at least one bid exists. Given time constraints, not all servers may send timely bids for several reasons (e.g., network errors). Thus, clients may not receive the optimal bid on every auction, but receive a sub-optimal bid that meets requirements. Still, in this paper we did not leverage on this requirement to get a better comprehension of the full operation of this auction in VFC.

Latency can represent the valuation/cost of clients and servers as it holds the main requirements for fog networks. For clients, it represents the cost they have on participating in the fog stratum. For the server, it represents well their internal costs, especially the en-queuing of tasks. The lower the latency, the higher the valuation is. Eq. (3) represents this relationship as clients only tend to participate in auctions when their valuation to process their tasks V_c is inferior to the valuation to use the fog services V_{fog}.

$$V_c > T_{\Delta t} + V_s \tag{3}$$

The objective of the auction is to minimize the service latency for clients. Besides strict latency requirements and evaluations, we aim to find the best scenario according to social welfare properties by considering the following functions:

- *Utilitarian Social Welfare* : $u(C, S) = \sum_{c,s} V_{c,s}$
- *Egalitarian Social Welfare* : $e(C, S) = min_{c,s} V_{c,s}$
- *Elitist Social Welfare* : $el(C, S) = max_{c,s} V_{c,s}$

The utilitarian social welfare function can provide a suitable metric for overall (or even the average) profit in a range of VFC applications, the egalitarian function suggests a level of fairness and may be a good performance indicator when

Table 2
Parameters for experiments.

Parameter	Value
Map	
Map	Manhattan 300 m (side)/ 150 m block
Vehicle density (veh/km^2)	33, 44, 111, 155
Simulation duration	100 s
Network	
Network throughput (Mbps)	27
Clients	
Client ratio	1/2; 1/3
Request rate (per seconds)	1, 10, 33.3, 100
Application type mean	100
Application type dispersion	1, 5, 10
Request type mean	100
Request type dispersion	1, 5, 10
Application load mean	1000.0
Application Load Dispersion	200.0
Request sizes (MB)	0, 300, 2197, 16695
Fog nodes	
Processing capacity	100000
Processing capacity dispersion	5000
Notification Method	1, 2

we have to satisfy the minimum requirements of a large number of clients/applications, finally, the elitist function can be useful in case of cooperation-based applications where we require only one client/application to conclude its processing.

At this point, we argue that such a mechanism tends to reduce overall service latency as several servers receive such notifications and cancel their tasks. The best-case occurs when the first server to send a bid is that with the shorter estimated processing time. On the other hand, the order of the servers sending bids affects the operation. Other servers may not cancel their tasks when the first server to send the bid has the highest estimated processing time. In this case, servers keep their tasks and thus present an increased processing redundancy as all servers keep processing tasks in parallel and do not reduce internal processing queues.

Also, the network is prone to errors that can cause some bids to get lost. Considering this nature and the similar behavior of both approaches, considering the nodes in the network as separated threads as m , and the tasks in a queue as n, the worst-case algorithmic complexity of such mechanisms is $O(m * n)$. The computational complexity is $O(m)$ if the operating system implements access to the application queue elements as a fast access method, such as hashing. That may reduce the complexity of searching enqueued tasks to delete during the auction.

4. Experimental evaluation

First, we investigate and model the internal behavior of a fog node (i.e., a server) under specific density, request rate and mobility conditions. As a simulation starts, the simulation core randomly selects a number of vehicles to become clients, and the remaining become servers (i.e., Fog Nodes) according to the parameter *Client Ratio*.[2] Then, the clients start generating requests resembling real applications. Fog Nodes in the vicinity receive those requests and negotiate the task processing allocation. We compare two resource management method types: (i) *auction-based* (described in Section 3) and (ii) *piggyback-based*. In the piggyback-based approach the servers do not send notifications/bids to the network after estimation. All servers process the task, and the fastest server broadcasts the response. Then, other servers receive the response and cancel the respective task. This mechanism generates processing redundancy but possibly reduces message redundancy.

4.1. Simulation setup

We leveraged on VEINS [26] to evaluate FogWise,[3] a state-of-the-art platform for simulating vehicular applications with fidelity. VEINS provides several extensions for accurate simulation of vehicular networks and applications, namely models of DSRC/WAVE lower layers, tailored propagation models, and representative vehicular scenarios.

Table 2 contains the relevant parameters for the evaluation. In the evaluation we consider an urban like scenario following the Manhattan grid pattern. The map size resembles the average coverage of RSUs or the vehicles' communication range. The considered simulation duration is sufficient for vehicles to enter and leave this map a few times at

2 The *Client Ratio* parameter is defined as the number of Clients related to the number of Servers.
3 https://bitbucket.org/ffmjunior/fogwise-veins/src/master/

the average speed of 14 m/s. We varied the client/server ratio and vehicle density to create scenarios according to the literature [5,17,27]. To generate different demand patterns, we simulate different vehicle densities ($33 - 155$ veh/km^2) and isolate this impact on network and communications.

"Application type" and "request type" are also essential parameters to model the heterogeneity environment in which fog nodes circulate. We also evaluated how Fog Nodes operate and how network behave when request sizes resemble real applications such as image and video processing: (i) *0 Bytes*: used as reference to represent a mock request with some identifiers for the server handling; (ii) *300 Bytes* for telemetry applications of vehicular networks [5]; and (iii) *2197/16695 Bytes* related to sizes of video frames with resolutions 144p and 480p, respectively. Those are common resolutions of videos applications that process images and videos (e.g. autonomous driving), or social networks.

FogWise models load and capacity in terms of their relative performance, i.e., there is one virtual processing unit per unit of time. An application will be processed in as many time units as needed according to its value, given its load. Hence, we get the run time [28]. We propose a measure of dispersion for the metrics *Processing Capacity Dispersion* and *Application Load Dispersion* to add a component of variability in the experiments. We argue that processing capacity and application load are subject to unknown variability in processing time, depending on the complex interactions among the server's computational components. The defined Processing Capacity and Application Load parameters indicate the vehicle can process real-time applications, such as object detection, and respect the requirements: about 33.3 ms for each frame in a video running at 30 fps, possibly including network latency.

4.2. Evaluation metrics

We performed experiments in an extensive range of metrics from VFC:

- **Queue Duration** [*Server*]: the average duration of the request queues in the simulation;
- **Number of Requests Received** [*Server*]: the average number of requests received by servers during simulation;
- **Number of Tasks Accepted** [*Server*]: number of tasks accepted after matching procedure;
- **Number of Tasks Finished** [*Server*]: number of tasks finished by a Fog Node after auction;
- **Number of Tasks Canceled** [*Auction*]: number of tasks canceled during the auction or piggyback-based methods due to estimated processing time from vehicles is shorter than the broadcasted bid;
- **Requests Overhead (RO)** [*Auction*] - measures duplicated responses when two or more Fog Nodes end up responding to the same request even after the auction finalization, which will be used to evaluate the effectiveness of the auction protocol.

The simulation framework generates these metrics for all vehicles, both clients and servers, and provides descriptive statistics. We perform factorial experiments with all combinations of parameters. The client and server metrics are collected and stored in logs for subsequent processing. The results in each plot represent an average of all vehicles and all simulations runs for a given scenario and a combination of parameters.

We also individualize the factors to identify their relevance within each scenario. We compare the samples that compose each point to calculate the differences using statistical tests. We leverage on the *t-test* when samples present normal distribution. On the other side, we leverage on the *Mann–Whitney* and *Wilcoxon* tests when samples require non-parametric statistics. The results show that all samples (i.e. points) isolated by factors have a statistically significant difference ($p < 0.05$) except for the difference between resource management methods (i.e., auction and piggyback-based methods).

We also ran a two-way ANOVA test to determine the influence of two factors (request size and density) on all response variables and evaluate whether there is an interaction between the factors. We evaluate the null hypothesis that groups identified by the response variable (e.g., a given metric) have an equal mean and that there is no significant interaction between the factors (e.g., request size, density). Table 3 contain the results of this assessment. We apply Monte Carlo permutation tests [29] on the data to relax the constraint of equal variance among the groups under the null hypothesis. The evaluation focuses on request size and density as the main factors that group results on plots. The obtained results indicate that most metrics present relevant differences when grouped by density and request size, with the notable exception of the metric *Number of Tasks Accepted*.

Table 3
p-values for ANOVA tests.

Metric	Factors	
	Request Size	*Density*
	p-value (Monte Carlo)	p-value (Monte Carlo)
Queue duration	0.0000	0.0000
Number of requests received	0.0000	0.0000
Number of tasks accepted	**0.5108**	**0.2736**
Number of tasks finished	0.0001	0.0000
Number of tasks canceled	0.0072	0.0000
Requests overhead	0.0000	0.0057

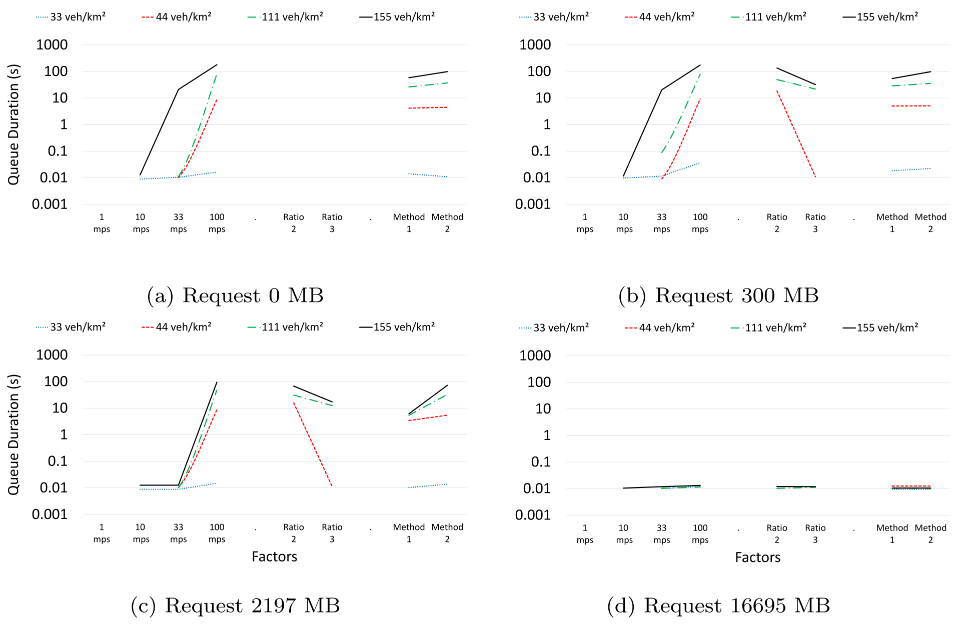


Fig. 2. Queue duration, in seconds, according to the emulated applications. Plots display request rate, client ratio and negotiation method.

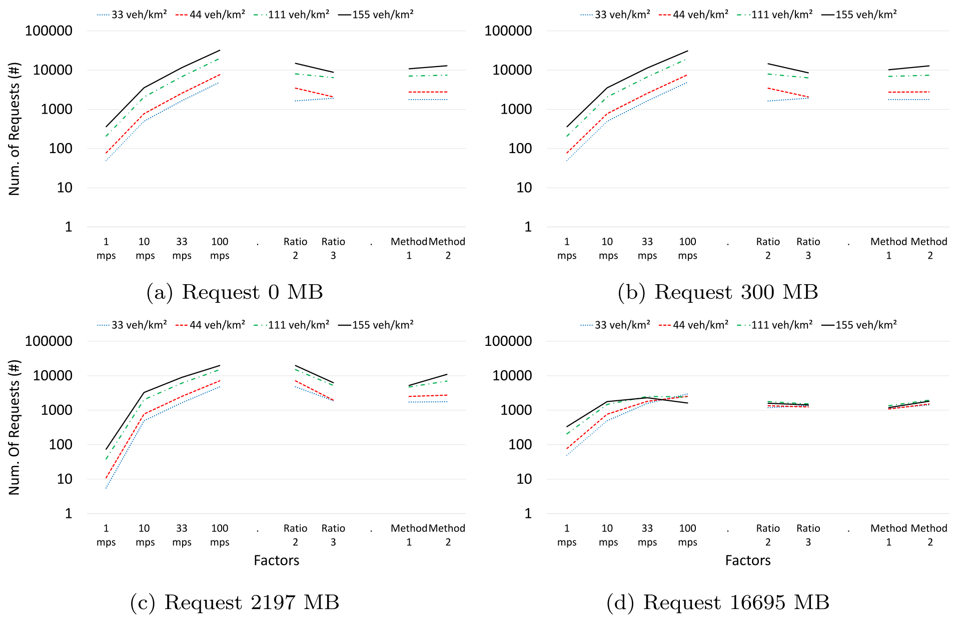
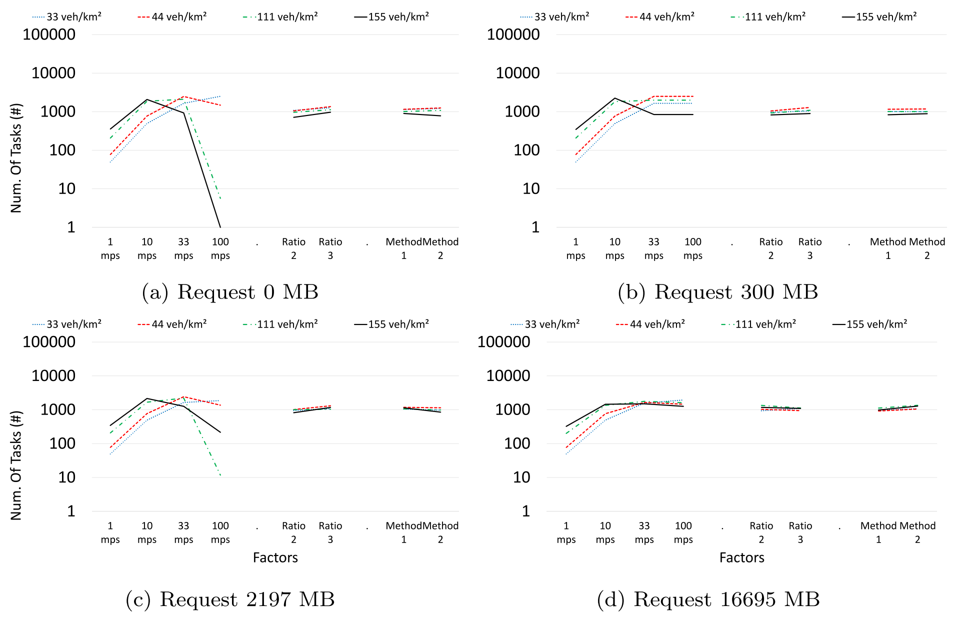


Fig. 3. Number of requests received according to the emulated applications. Plots display request rate, client ratio and negotiation method.

4.3. *Assessment of fog nodes*

Fig. 2 depicts the queue duration (in seconds) under different conditions. Each chart represents one message size with the data grouped by density. Results indicate that the queue duration increases with density and that this metric is higher for the piggyback method (*method 2*). In higher density scenarios, servers receive more requests thus the piggyback method makes servers keep redundant tasks for a long time until the fastest server responds. On the other hand, a lower client ratio reduces queue duration. Independently of the request size, the average queue duration remains close to 0 (zero) under the smallest density. For higher message sizes (2197 and 12695 Bytes), the queue duration presents a significant



(a) Request 0 MB (b) Request 300 MB

(c) Request 2197 MB (d) Request 16695 MB

Fig. 4. Number of tasks accepted according to the emulated applications. Plots display request rate, client ratio and negotiation method.

drop as the number of errors increases and the number of requests reaching the servers decreases. Also, we highlight that, differently from our previous work [8], there is no significant difference between proposed negotiation methods on the server side. This pattern holds also for the other server metrics.

Fig. 3 displays the results regarding received requests. This metric follows the same pattern as the queue duration, i.e. values increase with density and method, while dropping with the client ratio. Also, the values drop as the message size increases due to errors. The results from the statistical correlation tests indicate that the number of request could increase but is constrained by network errors. Also, Fig. 3(d) displays abnormal behavior caused by network errors as the number of requests received drops when messages are larger and density is higher. Figs. 3(a) and 3(b) have statistically related results displaying that request sizes up to 300 B work similarly with the baseline message size.

The number of tasks accepted, as displayed in Fig. 4, present patterns which are different from previous metrics due to the high influence of the request rate. At the highest request rate, the number of tasks accepted presents a significant drop, and this behavior is highly correlated with network errors. As the client ratio decreases, so the network errors also decrease and the number of tasks accepted increases. Regarding the negotiation method, the piggyback method presents an advantage in this metric under smaller message sizes. As the message size increases, the auction method makes servers accept proportionally more messages as redundant tasks are canceled and then those servers can process new tasks.

The number of tasks canceled, as displayed in Fig. 5, is crucial to evaluate negotiation methods. Initially, we observe that this metric is proportional to the number of tasks received. At lower densities, the number reduces with fewer clients. On the other hand, the value increases with the piggyback method, indicating lower levels of redundancy. This trend reverts to the largest message size as the network errors drastically reduce the number of requests accepted.

The number of tasks finished presented in Fig. 6, that also represents the number of responses by servers, indicates results after all steps of the auction mechanism. Similarly to other previous metrics, the number of tasks finished decreases for smaller densities at the highest message rate and largest request size. This metric displays extreme results, as displayed in Fig. 6(d). Also, at the largest message size, the piggyback method outperforms the auction. At smaller request sizes, the auction outperforms the piggyback method. Also in this case we observe the impact of message and network errors, especially when the auction performs negotiations before delivering the correct response.

4.4. Social welfare

The Social Welfare plots in Fig. 7 evaluate the fairness in the distribution of tasks and also reflect the other results. The lower the valuation/cost, the better for the auction is, i.e. in this scenario both clients and servers could receive/deliver services with reduced latency. All metrics present similar results when grouped by relevant parameters: density and message rate. By increasing the message rate, the distribution of the social-welfare i.e. the costs, increases in all scenarios. Results indicate a perceptual inflection point around 33 Messages Per Second (MPS) (third plot in each row), except when message size is 16695 B, which other results indicate anomalous behavior.

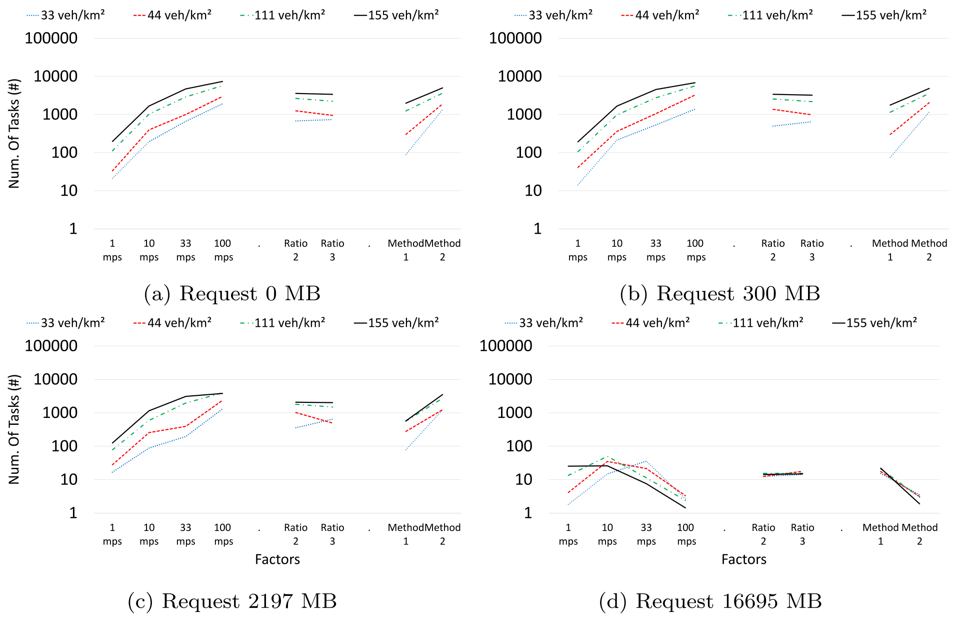


Fig. 5. Number of tasks canceled according to the emulated applications. Plots display request rate, client ratio and negotiation method.

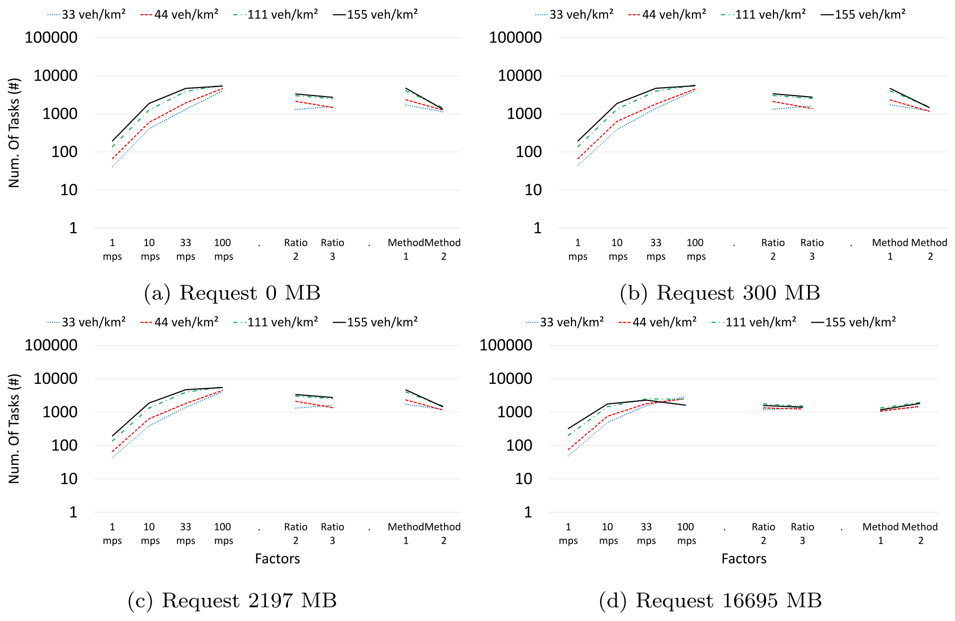
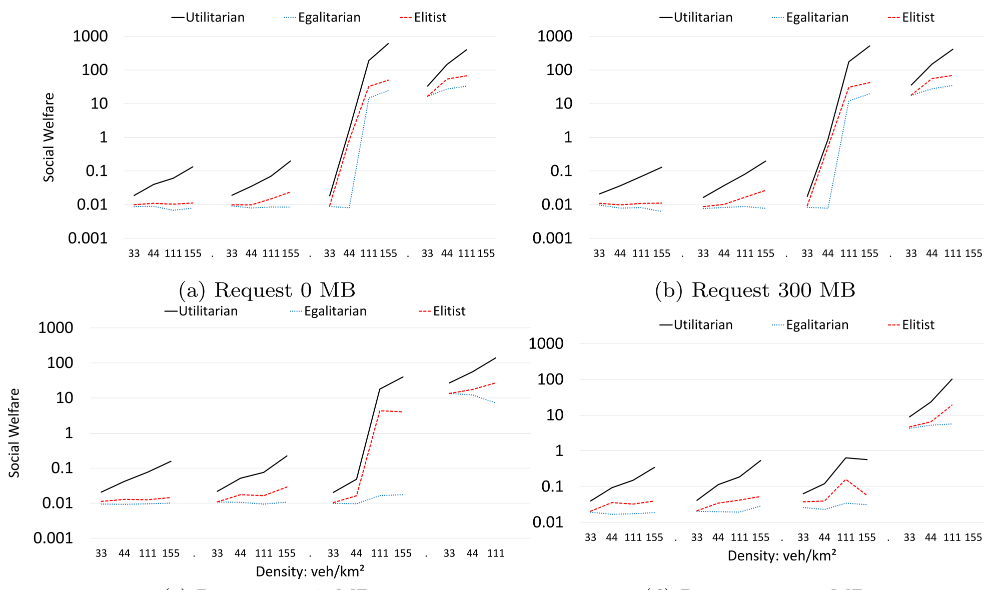


Fig. 6. Number of tasks finished according to the emulated applications. Plots display request rate, client ratio and negotiation method.

5. Conclusions

We performed an in-depth study of the operation of fog nodes under diverse network conditions and application requirements in an entirely distributed VFC. Our systematic literature review highlighted that the so far proposed approaches might not be suitable for latency-sensitive applications according to ITS requirements, as they lack evaluation at the network level and demonstration of latency constraints. We presented an operational model for VFC based on Open Descending First Price Auction, which we argue is suited for latency-sensitive applications as it delivers timely responses under certain conditions. The auction and the piggyback method deliver timely responses but present divergent results. The former generates message overhead while the latter generates processing overhead. Also, the network imposes limits



(a) Request 0 MB (b) Request 300 MB (c) Request 2197 MB (d) Request 16695 MB

Fig. 7. Utilitarian, Egalitarian and Elitist social welfare.

despite the idle or available processing power of fog nodes. Thus, we argue that fully distributed VFC cannot operate independently of Cloud and Edge Computing due to unpredictability in resource availability and network limitations.

VFC may run a small number of applications with limited message sizes and request rates or require high complexity levels in the algorithms to manage resource allocations and task assignments. Also, we argue that VFC is best suited for one hop-networking as relaying may increase the number of messages exchanged within the network, thus affecting applications' operation due to errors.

Statistical hypothesis tests using *Mann–Whitney*, *Wilcoxon* show that all samples have a statistically significant difference except in the case of resource management methods (i.e., auction and piggyback-based). Two-way Anova tests indicate that most metrics present relevant differences when grouped by density and request size except for the *Number of Tasks Accepted*.

The proposed framework could still be improved, e.g. by reducing the message overhead at the cost of some degree of complexity in the operation of fog nodes. Still, we argue that such complexity requires careful evaluation, as it may count as one more distributed VFC application. Such mechanisms also require robust security and privacy mechanisms, which were considered out of the scope for this work. Decentralized blockchain-based reputation and authentication mechanisms may emerge as reasonable solutions for security in VFC.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil - Research Productivity Fellowship - process number 12831/2020-0. This work was also partially supported by national funds of FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020).

References

[1] F. Yang, J. Li, T. Lei, S. Wang, Architecture and key technologies for internet of vehicles: A survey, J. Commun. Inf. Netw. 2 (2) (2017) 1–17, http://dx.doi.org/10.1007/s41650-017-0018-6.

[2] C. Zhu, G. Pastor, Y. Xiao, A. Ylajaaski, Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges, IEEE Commun. Mag. 56 (10) (2018) 58–63, http://dx.doi.org/10.1109/MCOM.2018.1800116.

[3] C. Zhu, Y.-H. Chiang, A. Mehrabi, Y. Xiao, A. Ylä-Jääski, Y. Ji, Chameleon: Latency and resolution aware task offloading for visual-based assisted driving, IEEE Trans. Veh. Technol. 68 (9) (2019) 9038–9048, http://dx.doi.org/10.1109/TVT.2019.2924911.

[4] S. Batewela, C.-F. Liu, M. Bennis, H.A. Suraweera, C.S. Hong, Risk-sensitive task fetching and offloading for vehicular edge computing, IEEE Commun. Lett. 24 (3) (2020) 617–621, http://dx.doi.org/10.1109/LCOMM.2019.2960777.
[5] X. Peng, K. Ota, M. Dong, Multiattribute-based double auction toward resource allocation in vehicular fog computing, IEEE Internet Things J. 7 (4) (2020) 3094–3103.
[6] T. Ouyang, Z. Zhou, X. Chen, Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2333–2345.
[7] J. Zhao, M. Kong, Q. Li, X. Sun, Contract-based computing resource management via deep reinforcement learning in vehicular fog computing, IEEE Access 8 (2019) 3319–3329.
[8] F.F. de Mendonça Junior, K. Lopes Dias, P.M. d'Orey, Z. Kokkinogenis, FogWise: On the limits of the coexistence of heterogeneous applications on fog computing and internet of vehicles, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) e4145.
[9] Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities, IEEE Wirel. Commun. 26 (1) (2019) 87–93.
[10] C. Xu, Y. Wang, Z. Zhou, B. Gu, V. Frascolla, S. Mumtaz, A low-latency and massive-connectivity vehicular fog computing framework for 5G, in: 2018 IEEE Globecom Workshops, GC Wkshps, IEEE, 2018, pp. 1–6.
[11] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez, Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach, IEEE Trans. Veh. Technol. 68 (4) (2019) 3113–3125.
[12] C. Zhu, G. Pastor, Y. Xiao, Y. Li, A. Ylae-Jaeaeski, Fog following me: Latency and quality balanced task allocation in vehicular fog computing, in: 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, IEEE, 2018, pp. 1–9.
[13] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, A. Ylä-Jääski, Folo: Latency and quality optimized task allocation in vehicular fog computing, IEEE Internet Things J. 6 (3) (2018) 4150–4161.
[14] F. Lin, Y. Zhou, G. Pau, M. Collotta, Optimization-oriented resource allocation management for vehicular fog computing, IEEE Access 6 (2018) 69294–69303.
[15] Y. Yao, X. Chang, J. Mišić, V. Mišić, Reliable and secure vehicular fog service provision, IEEE Internet Things J. 6 (1) (2018) 734–743.
[16] Y. Zhang, C.-Y. Wang, H.-Y. Wei, Parking reservation auction for parked vehicle assistance in vehicular fog computing, IEEE Trans. Veh. Technol. 68 (4) (2019) 3126–3139.
[17] Y. Kang, Z. Liu, Q. Chen, Y. Dai, Joint task offloading and resource allocation strategy for DiffServ in vehicular cloud system, Wirel. Commun. Mob. Comput. 2020 (2020).
[18] N. Cha, C. Wu, T. Yoshinaga, Y. Ji, K.-L.A. Yau, Virtual edge: Exploring computation offloading in collaborative vehicular edge computing, IEEE Access 9 (2021) 37739–37751.
[19] L. Yan, M. Zhang, C. Song, D. Wang, J. Li, L. Guan, Deep learning-based containerization resource management in vehicular fog computing, in: Asia Communications and Photonics Conference, Optical Society of America, 2019, pp. M4A–213.
[20] J. Klaimi, S.-M. Senouci, M.-A. Messous, Theoretical game approach for mobile users resource management in a vehicular fog computing environment, in: 2018 14th International Wireless Communications & Mobile Computing Conference, IWCMC, IEEE, 2018, pp. 452–457.
[21] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proc. of the ACM Workshop on Mobile Big Data, in: Mobidata '15, New York, NY, USA, 2015, pp. 37–42.
[22] B. Silva, W. Junior, K.L. Dias, Network and cloudlet selection for computation offloading on a software-defined edge architecture, in: Green, Pervasive, and Cloud Computing, Springer, 2019, pp. 147–161.
[23] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, ACM, 2013, pp. 15–20.
[24] 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.
[25] M. Volter, M. Kircher, U. Zdun, Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware, John Wiley & Sons, 2013.
[26] C. Sommer, R. German, F. Dressler, Bidirectionally coupled network and road traffic simulation for improved IVC analysis, IEEE Trans. Mob. Comput. 10 (1) (2011) 3–15.
[27] L. Rivoirard, M. Wahl, P. Sondi, M. Berbineau, D. Gruyer, Performance evaluation of AODV, DSR, GRP and OLSR for VANET with real-world trajectories, in: 2017 15th International Conference on ITS Telecommunications, ITST, IEEE, 2017, pp. 1–7.
[28] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.
[29] M. Anderson, C.T. Braak, Permutation tests for multi-factorial analysis of variance, J. Stat. Comput. Simul. 73 (2) (2003) 85–113.