future internet

MDPI

Article

Dynamic Allocation of SDN Controllers in NFV-Based MEC for the Internet of Vehicles

Rhodney Simões *, Kelvin Dias and Ricardo Martins

Centro de Informática, Universidade Federal de Pernambuco, Recife 50740-560, PE, Brazil; kld@cin.ufpe.br (K.D.); rmas@cin.ufpe.br (R.M.)
* Correspondence: rambks2@cin.ufpe.br

Abstract: The expected huge amount of connected cars and applications with varying Quality of Service (QoS) demands still depend on agile/flexible networking infrastructure to deal with dynamic service requests to the control plane, which may become a bottleneck for 5G and Beyond Software-Defined Network (SDN) based Internet of Vehicles (IoV). At the heart of this issue is the need for an architecture and optimization mechanisms that benefit from cutting edge technologies while granting latency bounds in order to control and manage the dynamic nature of IoV. To this end, this article proposes an autonomic software-defined vehicular architecture grounded on the synergy of Multi-access Edge Computing (MEC) and Network Functions Virtualization (NFV) along with a heuristic approach and an exact model based on linear programming to efficiently optimize the dynamic resource allocation of SDN controllers, ensuring load balancing between controllers and employing reserve resources for tolerance in case of demand variation. The analyses carried out in this article consider: (a) to avoid waste of limited MEC resources, (b) to devise load balancing among controllers, (c) management complexity, and (d) to support scalability in dense IoV scenarios. The results show that the heuristic efficiently manages the environment even in highly dynamic and dense scenarios.

Keywords: dynamic allocation of controllers; SDN; NFV-based MEC architecture; IoV; high-density scenarios

Citation: Simões, R.; Dias, K.; Martins, R. Dynamic Allocation of SDN Controllers in NFV-Based MEC for the Internet of Vehicles. *Future Internet* **2021**, *13*, 270. https://doi.org/10.3390/fi13110270

Academic Editors: Michael Sheng and Adnan Mahmood

Received: 30 August 2021
Accepted: 25 October 2021
Published: 26 October 2021

1. Introduction

The automotive industry has been evolved in a rapid pace and has also become an important player for the future hyper-connected (IoT) Internet of Things and (IoV) Vehicles [1]. As an icon of this industry, autonomous cars market is expected to reach USD 60 billion by 2030 [2]. The higher level of vehicle automation [3] implies an increased number of embedded sensors (e.g., radars, cameras) gathering and generating a massive amount of data [4]. The connected cars will rely not only on traditional V2I (Vehicle to Infrastructure) and V2V (Vehicle to Vehicle) communication modes, but also on V2X (Vehicular to Everything) one for connecting cars and the whole Intelligent Transportation System (ITS) to pedestrian, semaphores, as well as on cloud storage and processing for HD map services and decision making to e.g., prevent accidents, maneuver, lane change, and route optimization [5–9].

However, while some experiments have been carried out by players such as Waymo, Tesla, and Uber, they are not exploring the full potential of this paradigm due to the lack of holistic view of the ITS. In addition, decisions must be taken with very low latency in milliseconds with high reliability, a service categorized by International Telecommunication Union (ITU) for 5G (Fifth Generation) systems as URLLC (Ultra Reliable and Low Latency Communication) [10–12]. Thus, it becomes clear that not only vehicles' embedded sensors will suffice, but also networking, computing resources, overall environment knowledge and fast decision making also are of paramount importance to achieve the ultimate goal of

safety and self-driving through a full autonomic approach with no human intervention on the operation of IoV [13,14].

Recent efforts towards turning traditional and ossified VANETs (Vehicular Ad hoc Networks) into flexible/agile ones have been initiated by the adoption of Software-Defined Networking (SDN) to aid the management and control of vehicular networking [15–18]. SDN, originally devised in the context of data centers and wired networks, provides programmability by splitting the control and data planes. This way, network elements (data plane) rely on the intelligence of the SDN controller (i.e., logic that govern packet forwarding, dropping, and other control plane functionalities) running on commodity hardware equipment with global view to program network's elements. Upon arrival of novel flows at data plane, the SDN controller is in charge of defining rules to be installed in the network elements according to management policies implemented by the operator [19,20].

A first issue with regard to the IoV scalability has been addressed by adopting multiple SDN controllers to deal with the possible huge number of flow requests of vehicles [16,21]. These controllers are, in general, deployed on remote clouds, operator core network, or vehicular access points, which can be 802.11p/Road Side Units (RSUs), LTE eNBs (Long Term Evolution evolved Node B) or 5G gNBs, for example. However, even with the possibility of positioning the controllers at access points to achieve reduced communication delay, recent works still rely on static deployment [22]. This approach lacks processing capabilities and depends on the distant cloud. Thus, such solutions do not integrate the needed functionalities near the user for fast decision making. In summary, IoV faces critical requirements, which should be jointly designed: the scalable and dynamic deployment of SDN controllers and reduced communication latency [5].

Towards a flexible, scalable and dynamic deployment of SDN controllers, IoV scenarios could benefit from a highly complementary approach, namely Network Functions Virtualization (NFV) [23–25]. NFV aims at decoupling network functions (e.g., Firewall, MME, NAT, and load balancer) from the proprietary/dedicated hardware appliances to run as virtualized services in a cloud-based platform using general purpose equipment's (COTS) [26]. More specifically, NFV can benefit SDN, for example, by virtualizing SDN controllers as Virtualized Network Functions (VNFs), the software implementations of network functions, executed on a centralized server pool [27]. VNFs are provisioned and controlled through Management and Orchestration (MANO) Framework of NFV standard [28]. As far as we know, vehicular communications literature has no proposal on the synergy between SDN and NFV in order to address dynamic provisioning of SDN controllers in an NFV-based framework for Internet of Vehicles.

With regard latency bounds of IoV, a novel paradigm has also been recently emerged to compound the 5G and beyond conceptual architecture aiming to bring storage/computing capabilities requirements to the edge of the network, namely Multi-access Edge Computing (MEC) [29]. MEC has been standardized by European Telecommunications Standardization Institute (ETSI) since 2014 and provides cloud computing capabilities to the edge of the network through virtualized servers or micro datacenters at the eNBs, that is, within the operator's the Radio Access Network (RAN). Thus, MEC framework permits subscriber to offload tasks, faster communication of both data and control plane traffic, improving quality of experience by reducing latency, typically required for V2X scenarios [30–34]. MEC is mostly adopted for traditional IT cloud-based apps (e.g., video streaming, augmented reality), but it is still less commonly used for hosting network control plane functionalities.

To the best of our knowledge, there is no work on dynamic resource allocation of SDN controllers for the Internet of Vehicles leveraging on an NFV-based MEC infrastructure. This way, Dynamic Software Defined Vehicular Multi-access Edge Computing – DSDVMEC is propose as an autonomic extension to the ETSI MEC reference architecture along with ETSI NFV. DSDVMEC aims at being flexible and scalable through dynamically reducing the wasting of MEC resources due to previous allocations of SDN controllers that either became idle or not fully utilized by current demand, but without compromising the quality of service of ongoing flows coming from access points. In this way, DSDVMEC provides

the IoV context with the ability to scale the infrastructure of controllers as VNFs according to the variability of vehicle's flow on the roads, without compromising latency due to the proximity of SDN controllers located at the MEC. In scenarios where RSUs belong to different operators, they could present communication limitations when not using SDN. In addition, the absence of SDN controllers makes it difficult to have a global view of the network. Besides, the absence of SDN controllers makes it challenging to have a global view of the network. Due to this scenario, the highly variable demands from vehicles may degrade system management and scalability. To this end, an autonomic SDN Allocator is proposed as a component of the DSDVMEC in order to implement a heuristic taking into account: (a) scalability support, (b) resource usage optimization, and (c) management complexity of allocated controllers. In summary, the main contributions of this article are as follows:

- We propose DSDVMEC, an autonomic extension to the ETSI MEC-NFV architecture environment to dynamically allocate SDN controller for the Internet of Vehicles.
- An exact model to minimize the wasted resource utilization of SDN controllers in an MEC-NFV environment.
- A heuristic algorithm to handle highly dense and dynamic scenarios in IoV aiming at defining: (a) The required number of SDN controllers; (b) The assignment between RSUs and controllers; and (c) The amount of resources that should be allocated to the SDN controllers.

The remainder of this article is organized as follows. Section 2 discusses related work. Section 3 presents the DSDVMEC architecture. Section 4 formulate an exact model and the aSDNalloc heuristic responsible for the decision to allocate the controllers through VNFs. Numerical analysis, show the results and summarize possible directions for future research are presented in Section 5. Finally, Section 6 concludes this article. Table 1 is the summary of important acronyms for this paper.

Table 1. Summary of Important Acronyms.

Acronym	Definition	Acronym	Definition
5G	Fifth Generation	NFV	Network Functions Virtualization
aSDNalloc	autonomic SDN allocator	NFVO	Network Functions Virtualization Orchestrator
CFS	Customer Facing Service	NSGA-II	Non-dominated Sorting Genetic Algorithm II
DSDVMEC	Dynamic Software Defined Vehicular Multi-access Edge Computing	ONOS	Open Network Operating System
DSRC	Dedicated Short-Range Communications	PGA	Packing and genetic algorithm
eNB	Evolved Node B	PSO	Particle Swarm Optimization
ETSI	European Telecommunications Standardization Institute	QoS	Quality of Service
gNB	Next Generation Node B	RAN	Radio Access Network
GWO	Gray Wolf Optimizer algorithm	RRH	Remote Radio Head
ILP	Integer Linear Program	RSU	Road Side Unit
IoT	Internet of Things	SDMN	Software Defined Mobile Network
IoV	Internet of Vehicles	SDN	Software-Defined Network
ITS	Intelligent Transportation System	URLLC	Ultra Reliable and Low Latency Communication
ITU	International Telecommunication Union	V2X	Vehicle-to-everything
LCMA	Low-complexity controller master assignment	VANET	Vehicular Ad hoc Network
LTE	Long Term Evolution	VIM	Virtualization Infrastructure Manager
MANO	Management and Orchestration	VNF	Virtualized Network Functions
M-CORD	Mobile Central Office Re-architected as a Datacenter	VNFPRA	VNF placement and resource allocation
MEC	Multi-access Edge Computing	WAN	Wide Area Networks

2. Related Work

As the deployment of a single SDN controller to manage large scale scenarios is not an effective approach [35], distributed SDN controller architectures are unavoidable to tackle the huge volume of signaling and traffic of future dense vehicular networks. However, a challenge from the adoption of the distributed approach is how network devices should be assigned to the instances of SDN controllers. Such a NP-Hard optimization problem was initially defined as the Dynamic Controller Provisioning Problem in [36]. There exist various goals when approaching the analysis of such problem: minimizing the statistics collection cost, synchronization, assignment of switches to the controllers [36], minimizing the control flows, response time, and optimize the number of SDN controllers [37–39], maximizing the allocated resources and security [40,41]. Such goals can be seen in Table 2 and the pros/con in Table 3 also were analyzed in different contexts as discussed in the next subsections.

2.1. WAN

Seminal work on dynamic resource allocation of SDN controllers has its origins in the context of Wide Area Networks (WAN) [36]; such a reference presents the Dynamic Controller Provisioning Problem, an optimization strategy to minimize the cost of gathering statistics data from switches, configuration flows, synchronization, and large scale assignment of switch to controller. The authors formulated the optimal controller provisioning problem as an Integer Linear Program (ILP) and it was modeled as a Single Source Unsplittable Flow Problem. Since it is a NP-hard problem, two heuristics were proposed to solve it: a greedy approach based on the knapsack problem and a simulated annealing based meta-heuristic approach.

In [38] it is proposed a solution to the assignment of master controller to switches in a distributed SDN controller environment. While switches are connected to multiple SDN controllers for resiliency, only a master controller can install flow rules into switches. The master assignment strategy determines the master SDN controller for each switch. Thus, a master controller assignment problem is formulated in order to minimize the average setup flow latency taking into account the number of assigned master controllers along a path and the flow arrival rate among switches. A heuristic named low-complexity controller master assignment (LCMA) is presented to solve the problem.

2.2. Data Center

The work in [37] aims at minimizing the total cost due to the response time and maintenance of clusters of SDN controllers. To this, a two-phase algorithm is adopted. The goal of the first phase is to model the assignment problem as a stable matching with transfers. The second phase turns the matchings into connections between controllers and switches, which serves as input to a game theoretic strategy to achieve Nash equilibrium.

The proposals in [42] provide load balancing among SDN controllers. They are divided into centralized and decentralized approaches. However, centralized one is not feasible to be deployed in real environments, since it has limitations inherent to its architecture: single-point of failure and system bottleneck. On the other hand, the distributed approach is scalable, stable and more suitable for real environments. Its execution is divided into two phases: distributed startup and regionally balanced migration. In the initialization phase, switches are randomly assigned to a set of controllers. The second phase uses two variables: threshold and effluence. For this purpose, the number of output flows from the switch determines the weight of a switch, whereas the sum of all the weights of all switches assigned to a controller determines its weight at the time of measurement. In this way, the threshold is calculated using a factor that gives weight to past measurements, which is lesser than the weight of the current measurement multiplied by the average weight of all controllers. The authors adopted the term “effluence” to represent the result of the threshold multiplied by a factor greater than 1. If the controller load is less than or equal to the threshold, the controller is defined as being available to receive switches. However,

if the load is greater than the effluent then the controller is considered as overloaded and the switches must be assigned to other controllers marked as available. This work also presents how the solution would be implemented using the OpenFlow protocol.

The authors in [43] propose a heuristic approach to achieve an approximate solution within an acceptable time duration for the positioning of VMs in a balanced way considering the occupied and available servers resources (CPU, memory, and disk), intending to maximize the service rate and the index that determines the balance between VMs. The proposed heuristic uses a packing and genetic algorithm (PGA) based on Tetris and NSGA-II.

2.3. Wired Network

The maximization of existing resources is proposed in [40] without the need to allocate or release new resources through an approach based on switch migration. The proposal is modeled using Markov Chains and the algorithm has four stages: (1) the initialization, where the switches are assigned to existing controllers; (2) Next, it is selected a switch among the ones allocated to a specific controller. A count down timer is associated with this switch. When the timer expires, the switch will migrate to a neighbor controller. Before migration, its current controller will broadcast messages to all other controllers about the migration decision. The system status is updated with regard to the switches' utilization assigned to the controllers. (3) While the controller is waiting for the expiration of the count down timer, if messages from its neighbors arrive indicating it is a destination for future migrations, it will reset its count-down timer, as well as recalculates the switches' utilization and returns to the stage 2. (4) If the count-down timer expires, the switch will be migrated and the controller will broadcast messages to all other controllers, and return to stage 2. The proposal considers the monitoring of the CPU usage, memory, and bandwidth resources. The controller capacity is defined as amount of flows between controller and switch.

In [41] the authors devise a dynamic scheduling strategy for SDN controllers in order to maximize the security against attacks consisting of fake rules insertions into switches in a multi-controller environment. The proposal aims to avoid out of service due to network inoperability in scenarios of extreme conditions. The proposed security-aware and distributed [44] architecture is referred to as Mcad-SA, where M controllers are picked randomly out of the total N controllers at regular time intervals or upon a notification arrival at the scheduler. Since the probability of successful simultaneous attacks to the majority of controllers is low, controllers with similar flow rules are considered benign, while the others will be shut down. The optimization problem is formulated so as to guarantee (1) a reasonable switch cost by considering a threshold on total switch times of the set of current running controllers for each scheduling; and (2) the number of controllers is bounded by an upper value to avoid degradation of the QoS due to excessive delays incurred by the decision phase on rules' similarity if a large number of controllers were scheduled. As it is an NP-hard problem, the authors proposed a heuristic algorithm, MaxSG, which is composed of three steps: First, it assigns controller to switch, constrained by a maximum number of controllers in order to minimize the time for distribution of flow rules to switches. Next, the controllers are randomly selected and verification is conducted to guarantee the heterogeneity in terms of choosing controllers hosted by different machines, as well as distinct controllers' flavors (e.g., PoX, Floodlight, etc.). The previous step is then repeated until all selected controllers have been analyzed. Thus, the heuristic aims to use different flavors of controllers deployed in distinct machines to avoid insertion of fake flow rules, since a bug of a controller will not affect the others and also an attack to a machine will not compromise the others.

In [39] a multiple mapping approach between switches and controllers is proposed. A switch must be allocated to more than one controller in order to ensure resilience in a controller failure scenario. Therefore, the work aims to minimize the total time of configuration flows in the network considering the resilience constraint. This work models the controller response time through queuing theory. The authors assume that the proposed

approach provides stability to the analyzed environment, because if a controller fails, the switches linked to it did not suffer from delays, considering that other controllers will assume the task of assigning control flows, which does not happen in an environment where the switches are allocated to a single controller.

A dynamic controller association mechanism based on flow characteristics is proposed in [45] to reduce the resource utilization and traffic overhead. The mechanism is split into two stages. First, it is employed a fast algorithm based on greedy set coverage algorithm aiming at associating controllers to switches. Next, due to the unbalanced results of associations between controllers and switches from previous stage, migrations of switches are carried out through a coalitional game strategy to achieve the near-optimum association between controllers and switches.

2.4. Mobile Networks

The authors in [46] present a novel meta-heuristic named Gray Wolf Optimizer algorithm (GWO), which is based on Particle Swarm Optimization (PSO). This approach aims to solve the problem of dynamic allocation of SDN controllers, which is modeled as a problem of multiple backpacks, in the context of Software Defined Mobile Network (SDMN). To this end, GWO was inspired by the behavior of gray wolf packs to find a solution to a given problem. Thus, like the pack, the algorithm was divided into four groups: alpha, beta, delta, and omega. The first is the leader and represents the best solution. Beta and delta represent, respectively, the second and third best solutions and the remaining members are defined as omega. When the pack finds a prey, it creates a circle around it and gradually decreases the size of the circle until the prey cannot move, this behavior is abstracted to find the optimal solution to the problem. Initially, the GWO algorithm generates a random population and iteratively updates the population calculating the cost and defining the members of the wolf groups. To improve the performance of the algorithm and in order to avoid optimal locations, the chaotic map technique is used to generate deterministic random sequences. Assuming that each group is a set of controllers and switches, it was necessary to run two instances of GWO nested. The first level determines the best number of controllers for each group and the second level identifies the best connections between switches and controllers. The best solution between the two levels is to use the least amount of controllers connected to the switches.

In [47] is proposed a VNF placement and resource allocation (VNFPRA) problem. The authors devise as solutions two algorithms: a mixed integer program problem and a genetic based heuristic. Both solutions aim at reducing the overall placement and resource costs. The authors assume that just one VNF is not enough to meet the demands of all users' requests. Therefore, the replication of VNFs in other MEC nodes in the neighborhood is analyzed. A central controller with a global view of the network is used, which is responsible for deciding the optimal route for requesting VNFs. The results of the simulations indicate performance gains compared to solutions that already exist in the literature.

The authors in [48] propose an architecture for the automatic allocation of VNFs using important technologies and frameworks in an IoV scenario (M-CORD, XOS, ONOS, OpenStack, and OpenAirInterface). An algorithm for decision making regarding elasticity was devised based on VNF resources utilization as a parameter.

In [49] an extension of the MEC-NFV architecture for the allocation of SDN controllers is proposed. A testbed was implemented using OpenStack and Tacker to provide and manage the controller's life cycle. The main difference between [49] and this proposal is in the strategy adopted to decide the resources to be allocated to SDN controllers. In this proposal, the resources of each controller vary according to demand, whereas in [49] the controllers have the same amount of resources, changing only the number of controllers. In this proposal, a heuristic and an exact model are being demonstrated to optimize allocated resources on SDN controllers.

Table 2. Some of the Dynamic SDN controller assignment efforts in the literature.

Reference	Objective	Modeling	Strategy	Scenario	Evaluation
Bari et al. [36]	Minimize flow setup time and communication overhead	Single Source Unsplittable Flow Problem	Integer Linear Program	WAN	Simulation
Wang et al. [37]	Minimize the total cost caused by response time and maintenance on the cluster of controllers	Stable matching problems with transfers	-	DC	Simulation
Suh et al. [38]	Minimize the average flow setup latency	-	Integer Nonlinear Programming	WAN	Simulation
Sridharan et al. [39]	Minimize flow setup time	-	-	Traditional Network	Numerical analysis
Ye et al. [40]	Maximize control resource utilization	Switch Migration Problem	Markov Chain	Traditional Network	Prototype
Qi et al. [41]	Maximise security	-	-	Traditional Network	Simulation
Gao et al. [42]	-	-	Linear Programming	DC	Simulation
Wenting et al. [43]	Maximize the service rate for virtual machine placement	Bin packing problem	-	DC	Simulation
Li et al. [45]	Minimize control resource consumption and control traffic overhead	Resource optimization problem	Graph Theory	Traditional Network	Simulation
Farshin et al. [46]	-	Multi Knapsack Problem	Markov Chain	Software-Defined Mobile Networking	Simulation
Kiran et al. [47]	Minimize the overall placement and resource cost	VNF placement and resource allocation problem	Mixed Integer Program	MEC Network	Simulation
Mehmood et al. [48]	-	-	-	5G	Testbed
Sabino et al. [49]	-	-	-	5G-MEC	Testbed
Liu et al. [50]	Maximizing bandwidth efficiency and enhancing system scalability	Coding-Assisted Broadcast Scheduling Problem	Memetic Algorithm	SDN Vehicular Networks	Simulation
Dai et al. [51]	-	Cooperative Temporal Data Dissemination Problem	Classical Knapsack Problem	SDN Heterogeneous Vehicular Networks	Simulation

Table 2. *Cont.*

Reference	Objective	Modeling	Strategy	Scenario	Evaluation
Proposed solution	Minimize wasted resources considering load balancing between controllers, management complexity, and scalability	Bin Packing Problem	Linear Programming	IoV	Simulation

Table 3. Comparative of pros and cons between this proposal and previous works.

Reference	Pros	Cons
Bari et al. [36]	The first work using Integer Linear Program to optimal controller provisioning problem.	The switches may be assigned randomly among the activated controllers.
Wang et al. [37]	Since the proposal relies on the controller's worst response time, it will be possible to check if the designation choice meets the latency restrictions of the applications.	After matching, the strategy is that no switch will change its controller, which may not be useful in a context of high variability.
Suh et al. [38]	Consider the configuration time of switches with several controllers.	The proposal was not designed to be executed in real-time.
Sridharan et al. [39]	Two different approaches are proposed that aim to guarantee resilience considering the flow setup time.	It is assumed that all controllers have the same capacity.
Ye et al. [40]	It considers different imbalance metrics to maximize the use of resources.	It does not consider the change in the number of controllers.
Qi et al. [41]	Assume that when using different types of controllers, there is an increase in security.	Disregards the consequences of employing different controllers in the same architecture, such as increasing management complexity and limiting communication between controllers.
Gao et al. [42]	The proposal aims to divide the network into several parts with each part assigned to a controller.	Each part of the network is not necessarily balanced and the task of reconfiguration is complex.
Wenting et al. [43]	Employed heuristics to achieve an approximate optimal solution with acceptable time duration.	Did not compare the idealized model with the proposed heuristic.
Li et al. [45]	When employing fog, the latency between device and controller is reduced.	The vehicle relay communications ignore interference.

Table 3. *Cont.*

Reference	Pros	Cons
Farshin et al. [46]	It uses bio-inspired computing for making of dynamic dynamic allocation of controllers.	It does not consider controllers with different capacities.
Kiran et al. [47]	It minimizes the resource cost and overall placement.	Considered just one controller for the entire architecture generating a single point of failure.
Mehmood et al. [48]	Created a testbed to assess the dynamic allocation of VNFs in view of their use through technologies from the 5G scenario.	No exact model, only an algorithm was proposed to optimize the use of allocated resources.
Sabino et al. [49]	The creation of a testbed for 5G and MEC in the context of IoV for dynamic allocation of SDN controllers.	The absence of a robust strategy for the allocation of SDN controllers.
Liu et al. [50]	Employed a memetic algorithm to efficiently solve the CBS problem in an SDN-Based Vehicular Networks context.	Did not consider distributed SDN controllers for tolerance failure.
Dai et al. [51]	The heuristic proposed is able to improve overall system performance by adaptively distributing broadcast tasks of each request among multiple interfaces.	It does not consider controllers in edge computing.
Proposed solution	It considers the use of several controllers with different capacities being allocated dynamically in the face of demand fluctuation.	It disregarded the positioning of the controllers.

The work in [50] aims at maximizing bandwidth efficiency and enhancing system scalability by an SDN-based service architecture for heterogeneous vehicular communication environments.They formulate a novel problem of Coding-assisted Broadcast Scheduling (CBS) considering vehicular caching and network coding.

In [51] is proposed an SDN-based architecture to enable unified management on heterogeneous network resources, formulated the cooperative temporal data dissemination problem and heuristic algorithm, which synthesizes dynamic task assignment, broadcast efficiency, and service deadline into priority design.

3. DSDVMEC Architecture

The ETSI began the standardization of MEC, originally named as Mobile Edge Computing, in 2014 with the aim of creating an open environment to bring processing, networking, and storage to the network edge, thus reducing the latency for applications execution issued by mobile end-users. In 2016, the Mobile term was renamed to Multi-access aiming to embrace heterogeneous networking (e.g., WiFi and Fixed accesses). Due to its physical decentralization and high incurred costs of still limited computing resources with logically centralized management (Figure 1), there is a need to enhance the system with autonomic functionalities targeting to reduce the idle and underutilized resources, which is crucial for the wide acceptance and deployment of MEC as a supportive infrastructure for Internet of Vehicles. To this end, this article proposes the DSDVMEC (see Figure 2), where multiple SDN controllers are dynamically deployed via VNFs in an MEC infrastructure with the goal of granting reduced delays for SDN control signaling once backhaul latency is eliminated, as well as optimizing the MEC management and control operations to efficiently serve the end-users in densely and highly demanding IoV scenarios.

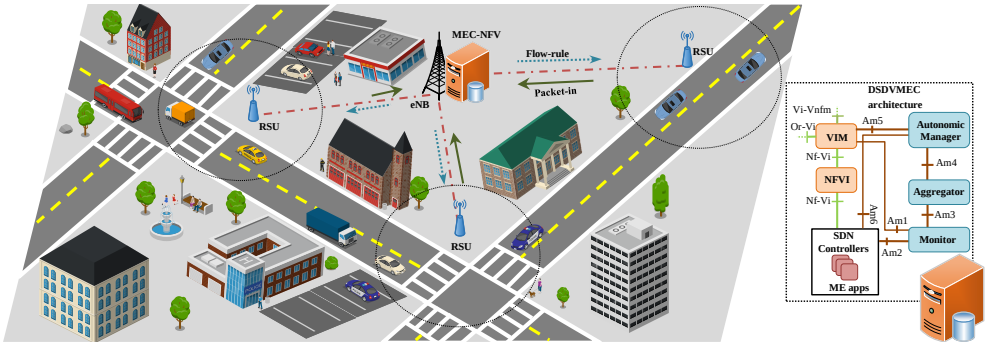

Figure 1. Scenario of IoV.

DSDVMEC dimensions the allocated resources as VNFs that run the SDN controllers and their allocations depend on the amount of requests from vehicles which are periodically monitored. This way, it is proposed an autonomic module in charge of self-management of the architecture through orchestration and resource optimization regarding the VNFs. To reduce the integration complexity and overload of existing functional blocks of the ETSI MEC reference architecture with NFV support [52], the autonomic module consists of three new functional blocks, namely, monitor, aggregator, and autonomic manager, besides four reference points, described as follows:

(1) Monitor: This block collects periodically the utilization of resources (CPU, memory, bandwidth, and storage) at Virtualization Infrastructure Manager (VIM) and the number of flows in controllers through an SDN monitoring application. (2) Aggregator: It is responsible for analyzing and summarizing the gathered data (e.g., the number of entrance flows interferes in the number of vehicles attached to eNBs and RSUs). The purposes of these actions are two-fold: correct errors during the collect phase and organize the data that serve as input to the heuristic-based decision making. (3) Autonomic Manager: It decides on the number of required controllers to meet a demand and also define which infrastructures will be assigned to the controllers as well as the computational resources to

be allocated to those controllers. Upon taking the decisions, this module checks whether existing controllers could be reused. This aims to avoid unnecessary operations for creating/releasing controllers. Next, it is also verified whether further controllers to meet the demand are necessary. At the end of the creation process, the assignment of infrastructure to controllers is carried out. Thus, the unstable time of the environment, i.e, the time between the demand arrival and the controllers assignment, is minimized. As the final step, the controllers not assigned to any infrastructure are released. The interaction between the autonomic module and the MEC-NFV reference architecture (Figure 2) occurs by means of the Monitor functional block, which gathers data from the VIM through the Am1 reference point and from the MEC applications through the Am2 reference point. In our proposal, MEC applications correspond to SDN controllers along with their applications provisioned by means of VNFs. Next, the data are sent to the Aggregator Module via Am3 reference point, that in turn, at the end of data treatment are delivered to the Autonomic Manager through Am4 reference point. The decisions taken by the Autonomic Manager with regard the creation/releasing of VNFs are transmitted through Am5 reference point to the VIM.

On the other hand, the assignment of communication infrastructures (eNBs and RSUs) to SDN Controllers is accomplished by an SDN application named Designator, which receives the decision taken by the Autonomic Module through the Am6 reference point.

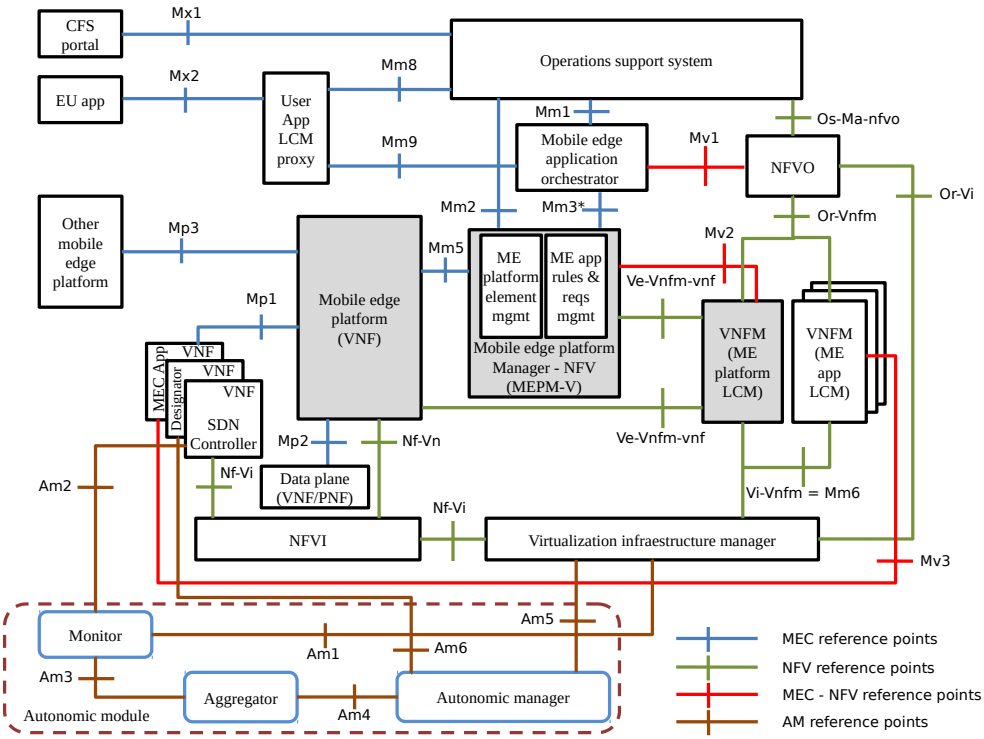

Figure 2. DSDVMEC architecture.

The decision and assignment/deployment process can be observed in Figure 3. It starts with the Monitor sending *Req. Reg. Flux* message requesting data about the flow-rules of SDN applications via Rest API. These applications forward the requests to the corresponding controllers using the Northbound Interface, which in turn request the infrastructure services using OpenFlow as the Southbound protocol to reply by sending back a *Resp. Reg. Flux* Message.

Then, the message *Req. Stat. MEC* is sent to the VIM indicating the request to use the computational resources of VNFs hosting the SDN controllers. The response message *Resp. Stat. MEC* is sent back to the Monitor.

Upon receiving the data, the Monitor forwards them to the aggregator for processing. Next, the Autonomic Manager runs the optimization heuristic to decide on resource allocation to controllers based on the data received from the Aggregator. After the heuristic

execution, detailed in the next section, the message *Req. Creat.VNF* to the VIM with the goal of creating new controllers along with their optimized corresponding computational resources. At the end of the process, the message *Resp. Creat.VNF* is sent to the Autonomic Manager. In the sequel, the message *Req. Sol. Assig* is sent to the SDN application, Designator, which is responsible to effectuate the assignment of communication infrastructures (eNb and RSUs) to the SDN controllers. The Autonomic Module receives the message *Resp. Sol. Assig*. when the allocation is concluded. Finally, the Autonomic Manager solicits to the VIM the exclusion of the controllers not assigned to infrastructures via message *Req. Del. NFV*. By doing the exclusions at the end of whole process, the latency of flow-rules due to assignments are reduced since previous ongoing flows of such excluded controllers have already been switched to the new ones.

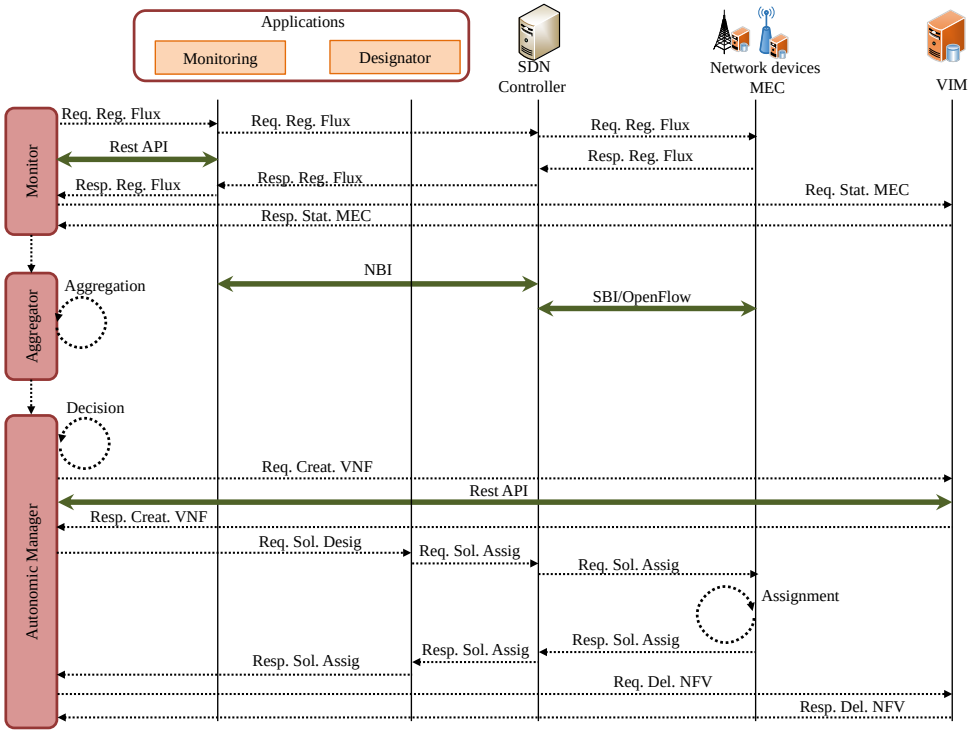

Figure 3. Sequence diagram for heuristic decision and assignment/deployment of SDN controllers.

4. Problem Formulation

In the proposed architecture depicted in Figure 1, the vehicular connectivity to the infrastructure is achieved through access points such as (IEEE 802.11p/DSRC) RSUs but is extensible to small cell or RRH (Remote Radio Head). For the sake of modeling, we considered that all RSUs have the same coverage and the MEC servers are located at the gNB. Each RSU is directly connected to the gNB site through a one-hop high-speed connection, forming an arrangement. The mobile core is not shown in Figure 1 for modeling purposes, our system considers reliable RSUs connections, but our architecture allows switching the connectivity to gNB when vehicles have both interfaces. Both RSUs and gNB are openflow-enabled, whereas the SDN controllers are instantiated at the MEC server.

We assume that there is sufficient capacity in the existing controllers to meet vehicle requests and that vehicles must first request content through an access point in order to acquire services. Considering there is no flow-rule to handle the request, the RSU issues a packet-in message to an SDN controller at the MEC to acquire a flow-rule. Then, the controller responds by sending a flow-mod message to the RSU to install a new entry in its flow-table. The *Monitor* periodically monitors the number of requests from the RSUs, D_j, and decides upon the assignment of RSUs to SDN controllers in order to balance the vehicular traffic/requests.

Furthermore, the RSUs will be connected to its arrangement's gNB through Ethernet links to reduce 5G connectivity costs. Let M be the set of RSUs; then each RSU j is responsible for receiving requests from vehicles, while D_j is the number of all requests from each vehicle to the RSU j. In this way, Y_i is a binary variable representing whether or not the controller i from N is picked. The controller capacity i is defined as C_i and the number of requests D_j from vehicles to RSUs assigned to a controller should not surpass the capacity C_i. Besides, a resource reservation factor of $0 \leq \omega \leq 1$ is granted to avoid overloading controllers during resizing periods. Table 4 defines the notations used in this letter. Figure 4 shows the flowchart of proposed model optimization approach.

$$Minimize: \sum_{i=1}^{N}(1-\omega)\cdot C_i \cdot Y_i - \sum_{j=1}^{M} D_j \tag{1}$$

$$S.T: \sum_{j=1}^{M} D_j \cdot X_{ij} \leq (1-\omega)\cdot C_i \cdot Y_i, \quad \forall i \in N, \tag{2}$$

$$\sum_{i=1}^{N} X_{ij} = 1, \quad \forall j \in M, \tag{3}$$

$$X_{ij} \in \{0,1\}, \quad \forall i \in N, \forall j \in M, \tag{4}$$

$$Y_i \in \{0,1\}, \quad \forall i \in N. \tag{5}$$

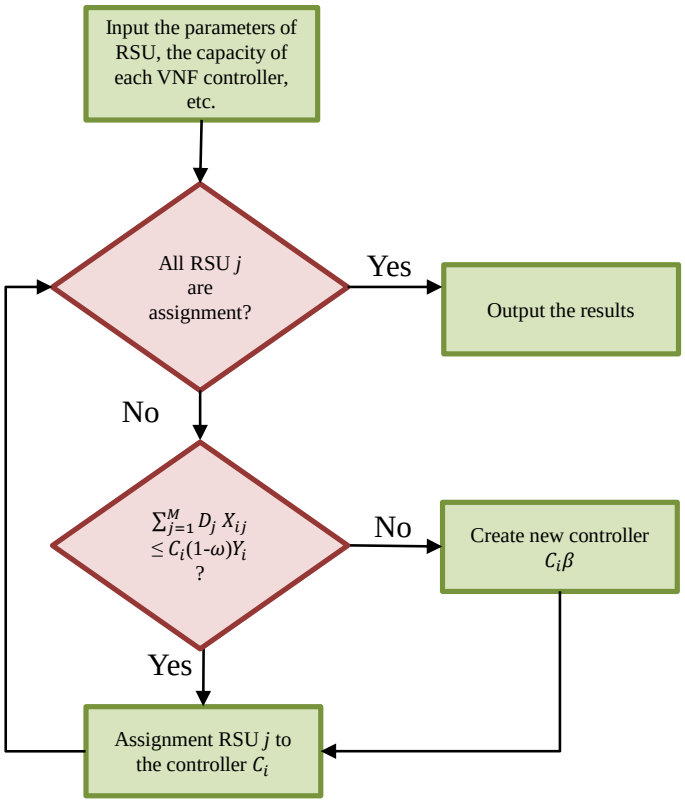

Figure 4. Flowchart of proposed model optimization approach.

Table 4. Notations.

Symbol	Definition
i, N	Index, set of VNF controllers.
j, M	Index, set of RSUs.
ω	Resource reservation factor
X_{ij}	Binary, X_{ij} = 1 if controller i is assignment to RSU j;0, otherwise.
Y_i	Binary, Y_i = 1 if controller i is chosen; 0, otherwise.
C_i	Capacity of each VNF controller i
D_j	Number of requests of vehicles to each RSU j
V	List of resource capacity templates

The objective function defined in Equation (1) aims to minimize the number of wasted resources; Constraint (2) ensures that the sum of the requests assigned to the controller does not exceed its capacity. Constraint (3) defines that an RSU is assigned to only one controller. Constraints (4) and (5) define the binary variables.

Heuristic

This section describes the proposed heuristic implemented by the Autonomic Manager named aSDNalloc - autonomic SDN allocator. aSDNalloc allocates dynamically computational resources to VNFs that run SDN controllers and applications. This solution is of fundamental importance to DSDVMEC since it reduces the amount of idle resources allocated to the controllers deployed in an MEC-NFV environment, which is not a resourceful data center. aSDNalloc was designed to run in shortest possible time without sacrificing the quality of the obtained solution. Since the execution time may directly impact on the response time for the infrastructure requests to the SDN controllers, aSDNalloc decision is based on three perspectives that must be taken into consideration by the heuristic, as described below:

- How many controllers should be required to meet the vehicular demand without incurring in overload or underutilization of their resources?
- How many computational resources (CPU, memory, bandwidth, storage) should be allocated to the SDN controllers?
- Which infrastructures (eNBs and RSUs) should be assigned to the SDN controllers?

The aSDNalloc was conceived based on the bin packing problem, since there is a need to reduce the number of required bins. In the context of SDN allocation, it means that the objective function will minimize the amount of allocated VNFs to meet the demand from communication infrastructures. This way, a heuristic takes the decision on the suitable number of controllers. aSDNalloc considers four input parameters, as described below:

1. List of resource capacity templates: a list: $V = v_1, v_2, ..., v_n$ represents the capacities for each template describing the computational resources created in the MEC in order to allocate SDN controllers. This list aims to provide the necessary information for the heuristic to decide about the most suitable set of resources so that controllers can adequately support each infrastructures' demand. As an example, consider the input $[x, z]$ meaning that the template x has the capacity to meet a demand z. Thus, aSDNalloc can group a set of n infrastructures by assuming that the demand of such set must be less or equal than the capacity z.
2. List of requests by RSU: $D = d_1, d_2, ..., d_m$ is a set of all RSUs, eNBs and their corresponding demands to be supported by the SDN controllers. This input is described as $[a, t]$, where a is the infrastructure identifier and t its respective number of requests.
3. ω: Threshold for the maximum utilization of VNFs. It aims to avoid resource starvation due to excessive utilization of computational resources, which could degrade

the QoS. Thus, aSDNalloc allows the definition of the amount of allocated resources that should be utilized to support a demand. For example, in a case where $\omega = 0.9$, 90% of the resources can be used, while 10% will correspond to a reservation to grant QoS and amortize the demand due to the vehicle mobility among communications infrastructures.

4. β: defines a multiplier for the infrastructure load in order to determine the minimum capacity of the controller to meet a demand. This parameter aims at reducing the number of SDN controllers, hence, diminishing the management complexity for the environment. To this end, a hypothetical example where $\beta = 10$ should be interpreted in the following way: In the case of there is no already allocated controller with available resources to meet the demand of a certain infrastructure, a new controller should be created with a minimum capacity 10 times greater than the demand for that infrastructure.

The pseudo-code is shown in Algorithm 1. aSDNalloc first assigns to d_{max} the RSU with the greatest number of requests from the list D, and to g_{wasted} the controller with the largest wasted capacity from list G (lines 3–4). Then, it checks whether g_{wasted} has enough spare capacity for supporting the number of requests from d_{max}. If this condition is satisfied, the RSU is assigned to the controller, which is inserted in its list G (lines 5–6). Otherwise, a new controller is created. To identify how many resources should be allocated to the new controller, the smallest resource v' is removed from the templates list V' (lines 7–11). Then, aiming at creating a controller with a minimum of β times the capacity required to serve d_{max} RSU, v' is compared with $(d_{max} \cdot \beta)$(line 12). The β parameter is a demand multiplier that aims to reduce the number of allocated controllers. Finally, a new controller $g_{v'}$ is created considering both the template v' and the number of reserved ω resources, as well as the assigned d_{max} (lines 13–14). These steps are repeated until there are no RSUs to be assigned.

Algorithm 1 Proposed Algorithm - aSDNalloc

Input: List of requests by RSUs, D
List of resource capacity templates, V
Demand multiplier, β
Amount of resource reservation, ω
Output: List of controllers with their resource capacity and respective assigned RSUs, G

1: $G = \emptyset$
2: **while** $R \neq \emptyset$ **do**
3: $d_{max} \leftarrow max_{request}(D)$
4: $g_{wasted} \leftarrow max_{wasted}(G)$
5: **if** $g_{wasted} > d_{max}$ **then**
6: $G \leftarrow G \cup$ **assignment**(g_{wasted}, d_{max})
7: **else**
8: $G \leftarrow G \cup g_{wasted}$
9: $V' = V$
10: **while** $V' \neq \emptyset$ **do**
11: $v' \leftarrow min(V')$
12: **if** $v' > d_{max} \cdot \beta$ **then**
13: $g_{v'} \leftarrow$ **create**(v', ω)
14: $G \leftarrow G \cup$ **assignment**$(g_{v'}, d_{max})$
15: **break**
16: **end if**
17: **end while**
18: **end if**
19: **end while**

In order to illustrate the idea behind the proposed heuristic, Figure 5 shows a simple example using the aSDNalloc. The first step consists on removing the greatest demanding

infrastructure from the list, which in turn must be allocated to a controller with an idle capacity greater or equal to this infrastructure, but respecting the constraint ω. If there is no such controller that meets these requirements, then, suitable resources from the list of capacities are assigned to the VNF in charge of executing the SDN controller. It is necessary to take into account that the minimum capacity must be greater than or equal to the infrastructure demand multiplied by the value of β. The abovementioned step is repeated until there is no infrastructure to be assigned to VNFs. It is possible to observe that in the third iteration the RSU with three requests is designated to the VNF-2, which had already assigned itself the RSU with five requests and had four units of idle resources. At the end of the iteration the VNF-1 remained with two idle resources and with the same amount in reserve before the adopted ω. VNF-2, on the other hand, had one resource unit idle and another unit in reserve before the value defined for ω. Last, it is checked the wasted capacity for all VNFs currently executing controllers in order to identify any underutilization of controllers. To this, a comparison among utilized capacity, wasted capacity, and the list of capacities is carried out. That way, since all infrastructures have already been assigned, then it is needed to identify if there is any overprovisioned VNF. In the affirmative case, resource reallocation must be done. The quality of obtained solution is determined by the summation of controllers' idle capacity, which consists in achieving the value 0 (zero) if the solution is optimal. Thus, the closer to zero, the better the quality of the solution provided by aSDNalloc.

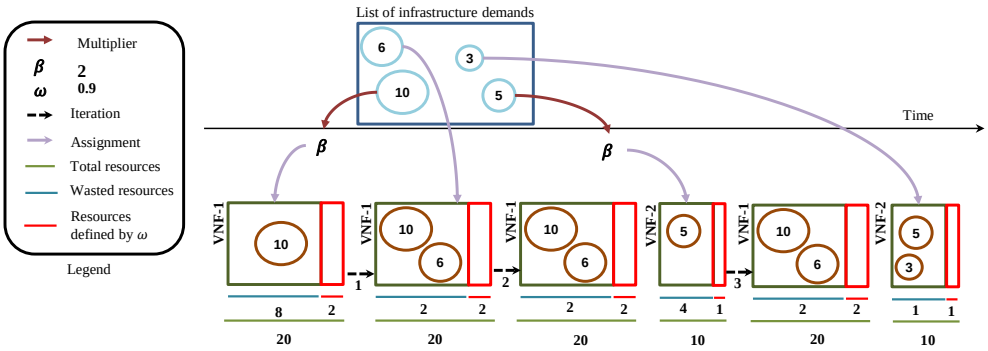

Figure 5. Heuristic process.

5. Experimental Analysis

We conducted numerous analyses taking into account two approaches, the analytical model using solver IBM ILOG CPLEX and heuristic written in Python. Two VANETs scenarios were defined: The first employed 50, 100, and 150 RSUs. It aimed at comparing the results of analytical model with the heuristic approach. The second is a high-density scenario with 5000, 10,000, and 15,000 RSUs, which, due to the intractable time required by the analytical model, considered only the heuristic.

The first and second scenarios adopted 500 and 10,000 distinct vehicle density instances, respectively. Each instance represents a new RSUs monitoring process carried out by the *Monitor* module, with each vehicle being responsible for a request, and the sum of the requests determining the RSU demand. Similarly to [53], vehicle mobility is considered in our experiments by means of scenarios with different vehicle densities.

The main goal of this experiment is to verify the applicability of using such heuristic for real-time decision-making in high-density scenarios. For both scenarios, the parameter β assumed the values 1, 10, 50, and 100. The RSU coverage radius was set to 250 m. Considering the capacity of the controller as ∼7500 requests per second [37], we adopt the templates capacities as 100, 500, 1000, 2000, 4000, 8000, 10,000, 12,000, and 15,000. The vehicles' density was chosen randomly within the interval [0, 250] vehicles/km as in [53]. For all experiments, the resource reservation parameter ω is set to 10%, and the graphs plotted with a confidence interval of 95%.

5.1. Wasted Resources

The first two analyses depicted in Figure 6 and Table 5 aim to verify how efficient the heuristic is through comparing it with the exact model and the efficiency of heuristics in a high-density scenario, respectively. Figure 6 illustrates that with the values of $\beta = 50$ and 100, there is an increase in wasted resources, but a reduction on the number of SDN controllers.

Adopting $\beta = 1$, the heuristic approach allocated more resources to controllers than the exact model. However, as the number of RSUs increases, the heuristics become more efficient, at worst-case less than 15% of resources wasted. In dense scenarios, the heuristic outcome improves, having analyzed in all cases less than 1% of wasted resources. It is noteworthy that all results in Table 5 had the CI variation less than 0.02%.

Table 5. Average capacity utilization in high density scenario.

Solution	Number of RSUs		
	5000	**10,000**	**15,000**
Alloc-1	99.34	99.42	99.46
Alloc-10	99.98	99.99	99.99
Alloc-50	99.95	99.98	99.99
Alloc-100	99.95	99.97	99.98

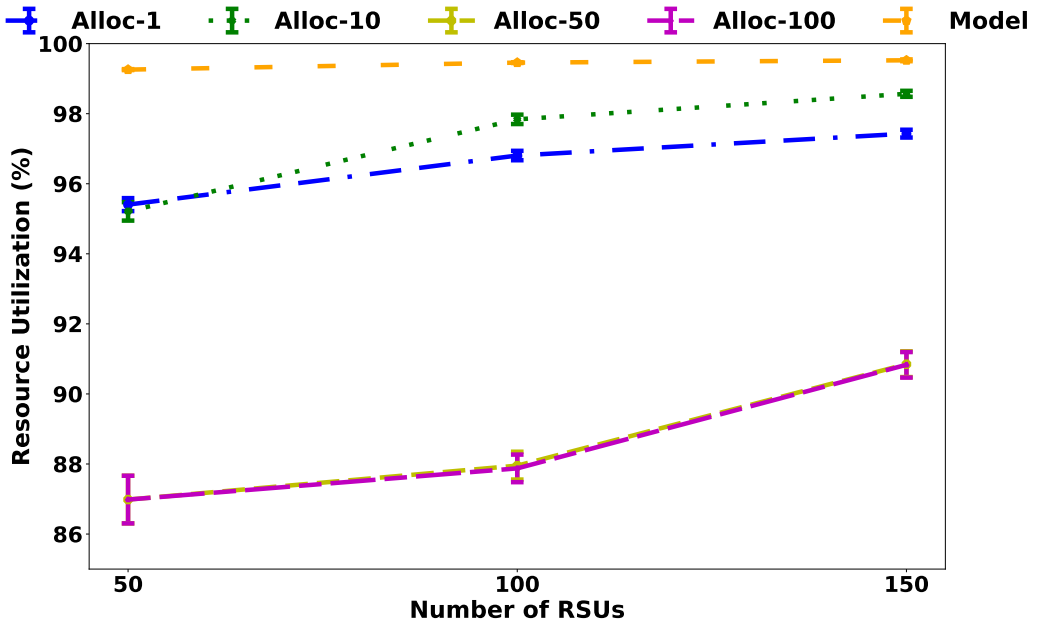

Figure 6. Capacity utilization.

5.2. Load Balancing

We considered the expression $\Delta_{Dif} = \Delta_{max} - \Delta_{min}$ [54] to verify the load balancing of the proposed solution, where $\Delta = \left(\frac{capacity-load}{capacity}\right)$, Δ_{max} is the biggest difference, and Δ_{min} the smallest one. Results vary in the range [0, 1], the closer to 0, the better the result, while the opposite, that is, the value 1, corresponds to the worst load balancing case. Table 6 shows that the heuristic becomes more efficient regarding both resource allocation and load balancing as vehicle density increases regardless of the value adopted for the β parameter in the first scenario.

In the high-density scenario (Table 7), the result was 0.02 in the worst-case and 0.00 in the best-case, the same behavior of the first scenario. This result shows that the proposed

solutions are not only efficient in allocating resources, but also promote load balancing between allocated SDN controllers.

Table 6. Δ_{Dif} for first scenario.

Solution	Number of RSUs		
	50	**100**	**150**
Model	0.02	0.01	0.38
Alloc-1	0.13	0.09	0.08
Alloc-10	0.12	0.05	0.03
Alloc-50	0.49	0.27	0.24
Alloc-100	0.49	0.27	0.19

5.3. Management Complexity

As more controllers are required, the more complex the environment management becomes. For example, considering the scenario with 15,000 RSUs, a huge amount of flows will require synchronization among controllers, which may cause packet flooding and, consequently, the degradation of applications and network performance. Therefore, the β parameter plays a crucial role in avoiding an unnecessary allocation of SDN controllers and an increase of such complexity and synchronization overhead.

Table 7. Δ_{Dif} for high-density scenarios.

Solution	Number of RSUs		
	5000	**10,000**	**15,000**
Alloc-1	0.02	0.01	0.01
Alloc-10	0.00	0.00	0.00
Alloc-50	0.00	0.00	0.00
Alloc-100	0.00	0.00	0.00

It is worth mentioning that the increased number of controllers does not necessarily related to an increase in wasted resources as evidenced by the result presented by the model for 150 RSUs, where there was a reduction in the number of controllers compared to the experiment with 100 RSUs. This behavior is the result of the number of templates providing different capacities to the controllers as shown in Figure 7.

For the high-density scenario evaluation shown in Figure 8, the number of controllers changed significantly with the scale increase. However, the heuristic is more efficient, kept the percentage of wasted resources low, adopting templates with higher capacities, and higher values for β, consequently, the number of controllers will reduce.

5.4. Scalability

Considering the dynamic aspect of the VANET environment, it is essential to understand the trade-off between the required time to deploy the SDN controllers into the MEC and the effectiveness of the achieved solution for both the exact model and the heuristic. As shown in Figure 9, as the scenario increases from 50 to 150 RSUs, the exact model becomes intractable, as it requires more than 1.09×10^5 ms to obtain the optimum solution. On the other hand, as shown in Figure 10, in a scenario with 15,000 RSUs, the heuristic requires less than 5.04×10^3 ms for $\beta = 1$. In addition, as the parameter β increases to 100, the time reduces to 3.03×10^2 ms. The results show that to decrease the heuristic execution time, it's necessary to adopt β with higher values, aiming at reducing the number of template selection tasks for the controllers. However, in the specific case of β set to 50 and 100 with 15,000 RSUs the values are similar. This fact can be explained by the number of drivers created, considering that more time is required to process the heuristic by creating new drivers than assigning RSUs to existing drivers.

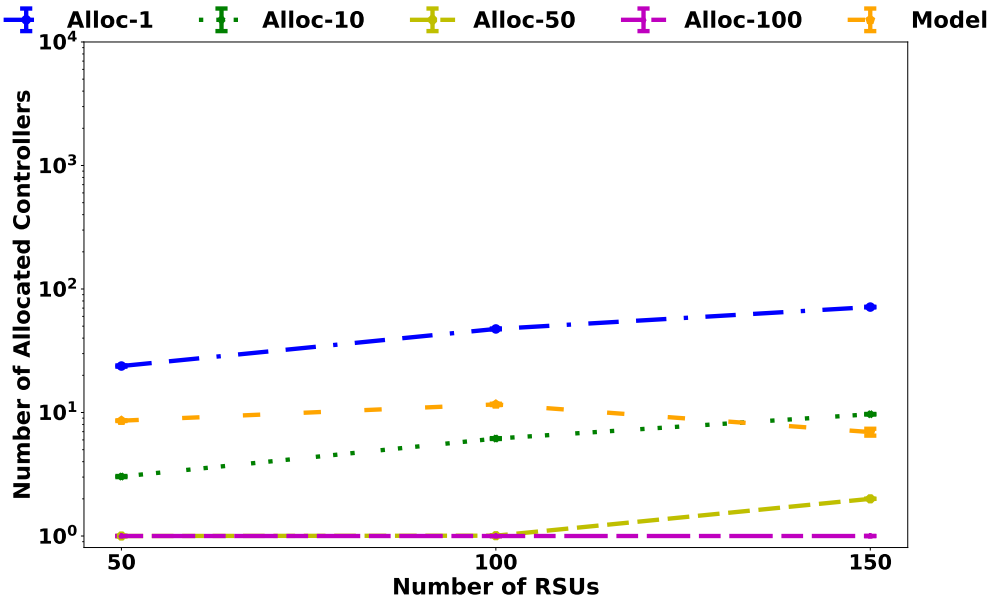

Figure 7. Numbers of allocated controllers.

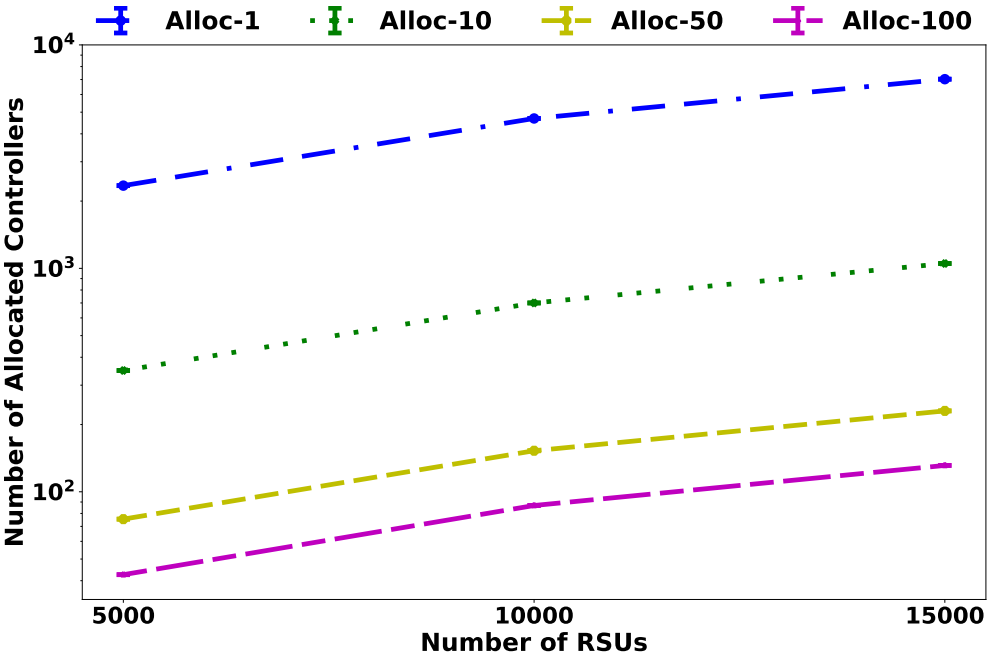

Figure 8. Numbers of allocated controllers in high-density scenario.

5.5. Limitations and Future Research Directions

In this work, the positioning problem of SDN controllers in IoV was not addressed, the choice of which RSUs will host which controller impacts the latency between the RSUs and the controllers, consequently, the time in which the flow-in is generated by the RSUs and the flow-rule is sent by the controller back to RSU. Although the results obtained by the heuristic evaluation are promising, the study still lacks evaluations with scenarios using real vehicular traces. We intende to evaluate our proposal with such real traces in future works. When adopting architectures that employ multiple controllers, one point to be considered is the communication necessary for synchronization between these controllers. One way to deal with this problem would be with east-westbound communication present in some controllers such as ONOS and OpenDaylight. It is also worth to mentioning that the increase in the number of controllers directly impacts the volume of control traffic. All abovementioned issues will be addressed in our future works.

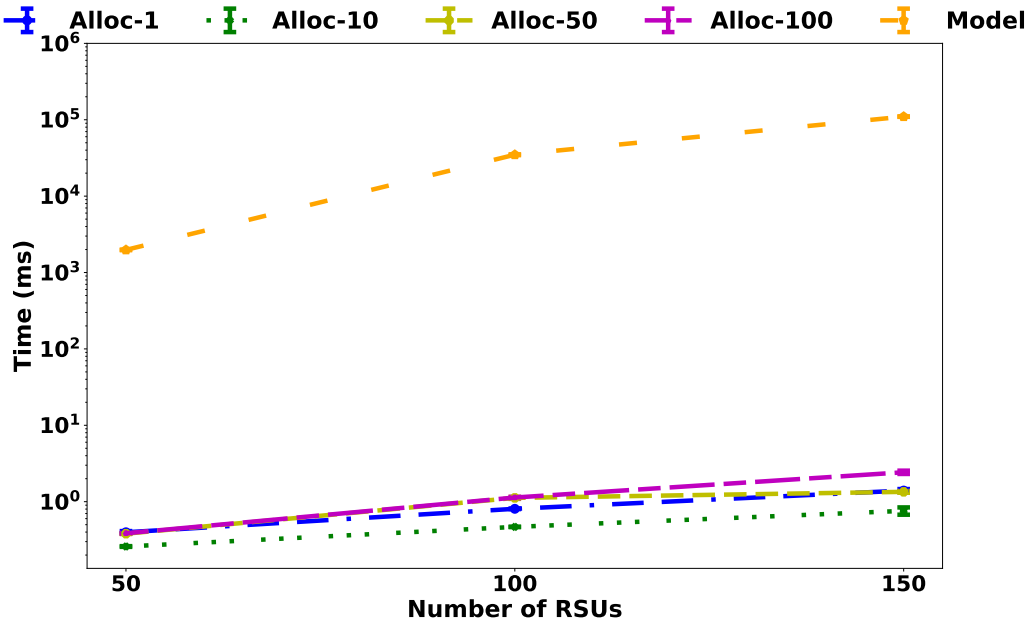

Figure 9. Average of time execution.

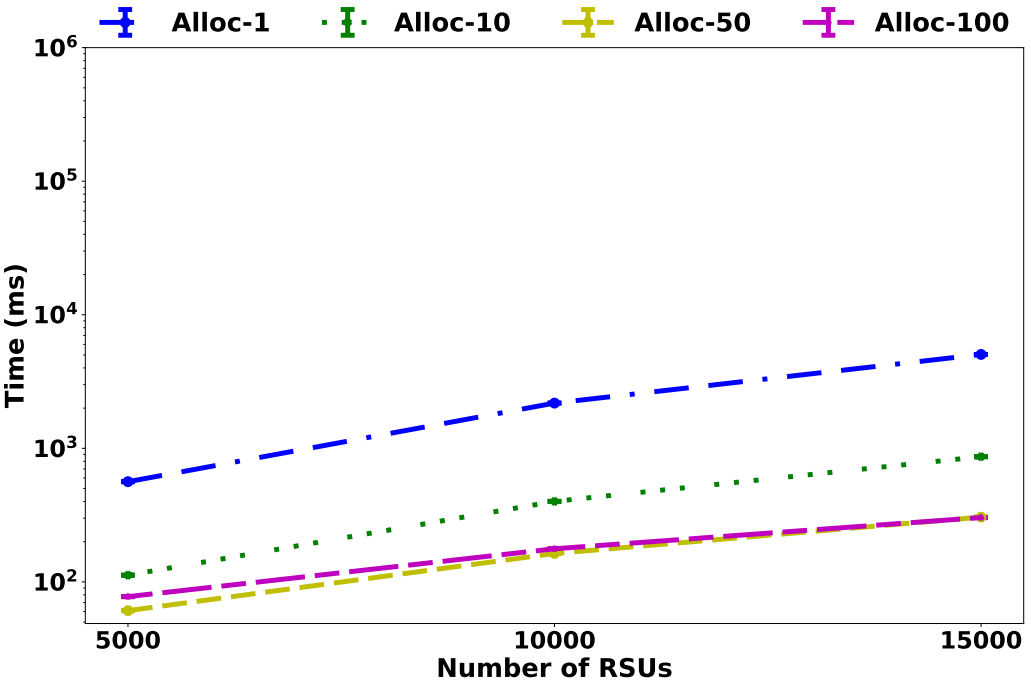

Figure 10. Average of time execution in high-density scenario.

6. Conclusions

In this article, we proposed a Dynamic Software Defined Vehicular Multi-Access Edge Computing solution in order to provide an agile and flexible architecture capable of meeting the requirements of applications in the context of the Internet of Vehicles. To this end, an MEC-NFV framework was devised to deploy dynamically SDN controllers as VNFs at the edge of the network. In addition, an exact model and the aSDNalloc heuristic were proposed to dynamically allocate SDN controllers for highly dense and dynamic IoV scenarios. The proposed heuristic provides a satisfactory tradeoff between reducing the complexity of the network infrastructure management and the volume of resources allocated to the VNFs that run SDN controllers. Bearing in mind that as the number of controllers is reduced, in order to reduce the complexity of managing the environment, the quality of the solution will get worse due to the increase of idle resources. aSDNalloc

obtained execution times less than 103 ms in highly dense scenarios with 15,000 RSUs and managed to reduce wasted resources to less than 1%. The next steps will be the implementation of a testbed for more accurate validation of the heuristic and the use of the remote cloud to allocate controllers responsible for inter-domain communication.

Author Contributions: Conceptualization, R.S.; Investigation, R.S.; Methodology, R.S.; Formal analysis, R.S. and K.D. and R.M.; Writing–original draft, R.S. and K.D; writing—review, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the FACEPE under Grant IBPG-0630-1.03/15 and CNPq grant number 312831/2020-0.

Data Availability Statement: Not Applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alouache, L.; Nguyen, N.; Aliouat, M.; Chelouah, R. Survey on IoV rout-ing protocols: Security and network architecture. *Int. J. Commun. Syst.* **2019**, *32*, e3849. [CrossRef]
2. United States Autonomous Car Market—Growth, Trends, and Fore-Cast (2019–2024). Available online: https://www.businesswire.com/news/home/20191223005261/en/United-States-Autonomous-Car-Market-Analysis-Competitive-Landscape-Growth-Trends-and-Forecasts-2019-2024---ResearchAndMarkets.com (accessed on 30 August 2021).
3. Zhang, S.; Chen, J.; Lyu, F.; Cheng, N.; Shi, W.; Shen, X. Vehicular Communication Networks in the Automated Driving Era. *IEEE Commun. Mag.* **2018**, *56*, 26–32. [CrossRef]
4. Xu, W.; Zhou, H.; Cheng, N.; Lyu, F.; Shi, W.; Chen, J.; Shen, X. Internet of vehicles in big data era. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 19–35. [CrossRef]
5. Zhou, H.; Xu, W.; Chen, J.; Wang, W. Evolutionary V2X Technologies Toward the In-ternet of Vehicles: Challenges and Opportunities. *Proc. IEEE* **2020**, *108*, 308–323. [CrossRef]
6. Liu, K.; Xu, X.; Chen, M.; Liu, B.; Wu, L.; Lee, V. A Hierarchical Architecture for the Future Internet of Vehicles. *IEEE Commun. Mag.* **2019**, *57*, 41–47. [CrossRef]
7. Butt, T.; Iqbal, R.; Shah, S.; Umar, T. Social Internet of Vehicles: Architecture and enabling technologies. *Comput. Electr. Eng.* **2018**, *69*, 68–84. [CrossRef]
8. Xu, W.; Shi, W.; Lyu, F.; Zhou, H.; Cheng, N.; Shen, X. Throughput Analysis of Ve-hicular Internet Access via Roadside WiFi Hotspot. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3980–3991. [CrossRef]
9. Abboud, K.; Omar, H.; Zhuang, W. Interworking of DSRC and Cellular Network Tech-nologies for V2X Communications: A Survey. *IEEE Trans. Veh. Technol.* **2016**, *65*, 9457–9470. [CrossRef]
10. Azari, A.; Ozger, M.; Cavdar, C. Risk-Aware Resource Allocation for URLLC: Chal-lenges and Strategies with Machine Learning. *IEEE Commun. Mag.* **2019**, *57*, 42–48. [CrossRef]
11. Pocovi, G.; Shariatmadari, H.; Berardinelli, G.; Pedersen, K.; Steiner, J.; Li, Z. Achiev-ing Ultra-Reliable Low-Latency Communications: Challenges and Envisioned System Enhancements. *IEEE Netw.* **2018**, *32*, 8–15. [CrossRef]
12. Abbasi, M.; Shahraki, A.; Barzegar, H.; Pahl, C. Synchronization Tech-niques in Device to Device- and Vehicle to Vehicle-Enabled Cellular Networks: A survey. *Comput. Electr. Eng.* **2021**, *90*, 106955. [CrossRef]
13. Ge, X. Ultra-Reliable Low-Latency Communications in Autonomous Vehicular Net-works. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5005–5016. [CrossRef]
14. Feng, D.; She, C.; Ying, K.; Lai, L.; Hou, Z.; Quek, T.; Li, Y.; Vucetic, B. Toward Ul-trareliable Low-Latency Communications: Typical Scenarios, Possible Solutions, and Open Issues. *IEEE Veh. Technol. Mag.* **2019**, *14*, 94–102. [CrossRef]
15. Duan, X.; Liu, Y.; Wang, X. SDN Enabled 5G-VANET: Adaptive Vehicle Clustering and Beamformed Transmission for Aggregated Traffic. *IEEE Commun. Mag.* **2017**, *55*, 120–127. [CrossRef]
16. Correia, S.; Boukerche, A.; Meneguette, R. An Architecture for Hierarchical Soft-ware-Defined Vehicular Networks. *IEEE Commun. Mag.* **2017**, *55*, 80–86. [CrossRef]
17. Silva Barbosa, F.; Mendonça Júnior, F.; Dias, K. A platform for cloudification of network and applications in the Internet of Vehicles. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3961. [CrossRef]
18. Ku, I.; Lu, Y.; Gerla, M.; Gomes, R.; Ongaro, F.; Cerqueira, E. Towards soft-ware-defined VANET: Architecture and services. In Proceedings of the 2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), Piran, Slovenia, 2–4 June 2014; pp. 103–110.
19. Bannour, F.; Souihi, S.; Mellouk, A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 333–354. [CrossRef]
20. Kreutz, D.; Ramos, F.; Veríssimo, P.; Rothenberg, C.; Azodolmolky, S.; Uhlig, S. Soft-ware-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]

21. Khan, A.; Abolhasan, M.; Ni, W. 5G next generation VANETs using SDN and fog computing framework. In Proceedings of the 2018 15th IEEE Annual Consumer Communications and Net-working Conference (CCNC), Las Vegas, NV, USA, 12–15 January 2018; pp. 1–6.
22. Liyanage, K.; Ma, M.; Chong, P. Controller placement op-timization in hierarchical distributed software defined vehicular networks. *Comput. Netw.* **2018**, *135*, 226–239. [CrossRef]
23. Gil Herrera, J.; Botero, J. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 518–532. [CrossRef]
24. Bonfim, M.; Dias, K.; Fernandes, S. Integrated NFV/SDN architectures: A systematic literature review. *ACM Comput. Surv. (CSUR)* **2019**, *51*, 1–39. [CrossRef]
25. Souza, R.; Dias, K.; Fernandes, S. NFV Data Centers: A Systematic Review. *IEEE Access* **2020**, *8*, 51713–51735. [CrossRef]
26. Li, Y.; Chen, M. Software-Defined Network Function Virtualization: A Survey. *IEEE Access* **2015**, *3*, 2542–2553.
27. ISG. Gs nfv-eve 005 v1. 1.1 Network Function Virtualisation (nfv); Ecosystem; Report on Sdn Usage in nfv Architectural Framework. Available online: https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_nfv-eve005v010101p.pdf (accessed on 30 August 2021).
28. Sousa], N.; Perez], D.; Rosa, R.; Santos, M.; Rothenberg], C. Net-work Service Orchestration: A survey. *Comput. Commun.* **2019**, *142–143*, 69–94.
29. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1657–1681. [CrossRef]
30. Wu, Y.; Zheng, J. Modeling and Analysis of the Downlink Local Delay in MEC-based VANETs. *IEEE Trans. Veh. Technol.* **2020**, *69*, 6619–6630. [CrossRef]
31. Liu, J.; Wan, J.; Zeng, B.; Wang, Q.; Song, H.; Qiu, M. A Scalable and Quick-Response Software Defined Vehicular Network Assisted by Mobile Edge Computing. *IEEE Commun. Mag.* **2017**, *55*, 94–100. [CrossRef]
32. Huang, C.; Chiang, M.; Dao, D.; Su, W.; Xu, S.; Zhou, H. V2V Data Offloading for Cellular Network Based on the Software Defined Network (SDN) Inside Mobile Edge Computing (MEC) Architecture. *IEEE Access* **2018**, *6*, 17741–17755.
33. Deng, Z.; Cai, Z.; Liang, M. A Multi-Hop VANETs-Assisted Offloading Strategy in Vehicular Mobile Edge Computing. *IEEE Access* **2020**, *8*, 53062–53071. [CrossRef]
34. Wu, Y.; Zheng, J. Modeling and Analysis of the Uplink Local Delay in MEC-Based VANETs. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3538–3549. [CrossRef]
35. Singh, A.; Srivastava, S. A survey and classification of controller placement problem in SDN. *Int. J. Netw. Manag.* **2018**, *28*, e2018. [CrossRef]
36. Bari, M.; Roy, A.; Chowdhury, S.; Zhang, Q.; Zhani, M.; Ahmed, R.; Boutaba, R. Dynamic controller provisioning in software defined networks. In Proceedings of the 2013 9th International Conference on International Conference on Network and Service Management (CNSM), Zürich, Switzerland, 14–18 October 2013; pp. 18–25.
37. Wang, T.; Liu, F.; Xu, H. An efficient online algorithm for dynamic SDN controller assignment in data center networks. *IEEE/ACM Trans. Netw.* **2017**, *25*, 2788–2801. [CrossRef]
38. Suh, D.; Pack, S. Low-Complexity Master Controller Assignment in Distributed SDN Controller Environments. *IEEE Commun. Lett.* **2018**, *22*, 490–493. [CrossRef]
39. Sridharan, V.; Gurusamy, M.; Truong-Huu, T. On Multiple Controller Mapping in Software Defined Networks With Resilience Constraints. *IEEE Commun. Lett.* **2017**, *21*, 1763–1766. [CrossRef]
40. Ye, X.; Cheng, G.; Luo, X. Maximizing SDN control resource utilization via switch migration. *Comput. Netw.* **2017**, *126*, 69–80. [CrossRef]
41. Qi, C.; Wu, J.; Hu, H.; Cheng, G. Dynamic-scheduling mechanism of controllers based on security policy in software-defined network. *Electron. Lett.* **2016**, *52*, 1918–1920.
42. Gao, X.; Kong, L.; Li, W.; Liang, W.; Chen, Y.; Chen, G. Traffic load balancing schemes for devolved controllers in mega data centers. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 572–585. [CrossRef]
43. Wei, W.; Wang, K.; Wang, K.; Gu, H.; Shen, H. Multi-resource balance opti-mization for virtual machine placement in cloud data centers. *Comput. Electr. Eng.* **2020**, *88*, 106866. [CrossRef]
44. Qi, C.; Wu, J.; Hu, H.; Cheng, G.; Liu, W.; Ai, J.; Yang, C. An intensive security architecture with multi-controller for SDN. In Proceedings of the 2016 IEEE Conference On Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, USA, 10–14 April 2016; pp. 401–402.
45. Li, Z.; Hu, Y.; Hu, T.; Wei, P. Dynamic SDN Controller Association Mechanism Based on Flow Characteristics. *IEEE Access* **2019**, *7*, 92661–92671. [CrossRef]
46. Farshin, A.; Sharifian, S. A chaotic grey wolf controller allocator for Soft-ware Defined Mobile Network (SDMN) for 5th generation of cloud-based cellular systems (5G). *Comput. Commun.* **2017**, *108*, 94–109. [CrossRef]
47. Kiran, N.; Liu, X.; Wang, S.; Yin, C. VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks. In Proceedings of the 2020 IEEE Wireless Communications And Net-Working Conference Workshops (WCNCW), Seoul, Korea, 6–9 April 2020; pp. 1–6.
48. Mehmood, A.; Muhammad, A.; Ahmed Khan, T.; Diaz Rivera, J.; Iqbal, J.; Ul Islam, I.; Song, W. Energy-efficient auto-scaling of virtualized network function instances based on resource execution pattern. *Comput. Electr. Eng.* **2020**, *88*, 106814. [CrossRef]

49. Sabino, A.; Simoes, R.; Dias, K. Dynamic Controller Instantiation in MEC-NFV Based Software Defined Internet of Vehicles. Available online: https://sol.sbc.org.br/index.php/sbesc_estendido/article/view/13098 (accessed on 30 August 2021).
50. Liu, K.; Feng, L.; Dai, P.; Lee, V.; Son, S.; Cao, J. Coding-Assisted Broadcast Schedul-ing via Memetic Computing in SDN-Based Vehicular Networks. *IEEE Trans. Intell. Transp. Syst*. **2018**, *19*, 2420–2431. [CrossRef]
51. Dai, P.; Liu, K.; Wu, X.; Yu, Z.; Xing, H.; Lee, V. Cooperative Temporal Data Dissem-ination in SDN-Based Heterogeneous Vehicular Networks. *IEEE Internet Things J*. **2019**, *6*, 72–83. [CrossRef]
52. Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV Environment. Available online: https://www.etsi.org/deliver/etsi_gr/mec/001_099/017/01.01.01_60/gr_mec017v010101p.pdf (accessed on 30 August 2021).
53. Ge, X.; Li, Z.; Li, S. 5G Software Defined Vehicular Networks. *IEEE Commun. Mag*. **2017**, *55*, 87–93. [CrossRef]
54. Khorramizadeh, M.; Ahmadi, V. Capacity and load-aware soft-ware-defined network controller placement in heterogeneous environments. *Comput. Commun*. **2018**, *129*, 226–247. [CrossRef]