EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Migration of Microservices Execution Contexts Between Processing Zones in Software-Defined Vehicular Fog Networks

Leonel D. C. Alvarenga[1,2,*], Pedro Sousa[2], António Costa[2]

[1]Instituto Federal Goiano, Rio Verde - GO, Brasil
[2]Centro Algoritmi, LASI, Department of Informatics, University of Minho, Braga, Portugal

Abstract

The growing need for time-sensitive applications in vehicular networks makes Fog Computing a promising model to orchestrate cloud-based services consumed by vehicle nodes, as it brings computational resources and decision-making processes near to vehicles. However, vehicular mobility presents significant challenges for implementing Fog Computing Services that require low latency. The dynamic nature of vehicle movement means that the physical location of computational resources (Fog Nodes) relative to vehicles is constantly changing. Consequently, maintaining a consistent and reliable low-latency communication path becomes challenging. Recent research suggests that the Software-Defined Networking paradigm can optimize Vehicular Fog Computing Networks in resource and service management. In this paper, a dynamic methodology is proposed for the migration of the microservice execution context between processing zones in Software-Defined Vehicular Fog Computing Networks (SDVFN). This approach was tested using a simulated use case in our SDVFN Simulation Framework, designed to support research on dynamic microservice orchestration in SDVFN, taking into account vehicular mobility.

Received on 26 September 2024; accepted on 07 March 2025; published on 19 August 2025

Keywords: Vehicular Fog Networks, Software-Defined Vehicular Fog Networks, Task Allocation, Mobility-Responsive Microservice Orchestration

doi:10.4108/eetiot.9987

1. Introduction

With the expansion of urban areas and the increase in the number of vehicles, applications of Intelligent Transportation System (ITS) are revolutionizing the management and interaction within vehicular networks. Typically, ITS refers to the application of information, communication, and sensing technology to transportation and transit systems. It is likely to be an integral component of the smart cities of the future [1].

In this context, Internet of Vehicle (IoV), under the umbrella of Internet of Things (IoT), emerges as a promising paradigm for enabling a wide variety of ITS applications, taking advantage of connectivity and data exchange capabilities between vehicles and infrastructure, and real-time sharing of information such as traffic conditions, road hazards, and weather updates. ITS can improve road safety and efficiency, optimize traffic flow, mitigate congestion, and improve the overall driving experience [2–5].

Vehicular nodes in the IoV paradigm can benefit from various services provided by Cloud and Fog Computing Networks. While Cloud Computing offers a centralized platform for processing and storing data, Fog Computing brings computational resources closer to the edge of the network, enabling low-latency processing and real-time decision-making. Being the junction of these two network paradigms, Vehicular Fog Network (VFN) is an environment in which vehicular nodes can access many services that are hosted on a Fog Computing Network. This is particularly beneficial for applications that require immediate responses. By handling data locally, VFN reduces the amount of data to be transmitted to the cloud, conserving

*Corresponding author. Email: leonel.carvalhaes@ifgoiano.edu.br

bandwidth, and reducing costs. It also improves service availability, allowing operations to continue even when the connection to the cloud is lost, and increases data privacy and security by minimizing data exposed to network transmission [6].

Within an IoV environment, Fog platforms need to face additional challenges as vehicles move between Road-Side Units (RSUs), while preserving their roles in computing, communication, and end-user functionality. Furthermore, the stringent Quality of Service (QoS) requirements of numerous ITS applications impose significant challenges for the deployment of VFN [2].

Modern networking paradigms, specifically the Software-Defined Networking (SDN), can be used to improve Fog Computing orchestration to manage resources and services across edge devices [7]. It can solve problems and limitations that occur in traditional environments by using an architecture designed to simplify and improve network management. In the SDN architecture, the control plane and the data plane are decoupled, so the control of the network is separated from the packet forwarding mechanism. Thus, the network control function can be programmed directly, adapting to the reality of each scenario, as well as promoting improvements in existing mechanisms [8].

SDN paradigm is applied to get better results on typical tasks in IoVs and vehicular ad hoc networks. This paradigm has become an excellent alternative to promoting centralized network control, improving flexibility and programmability, simplifying network management, optimizing wireless network resources, improving heterogeneity management and enabling even more intelligent and scalable services in Vehicular Network environments [9, 10].

The authors in [7] present research on SDN and Fog Computing in vehicular networks. They envision an architecture that uses SDN to support Fog Computing in Vehicular Ad Hoc Networks (VANETs). The authors do not deal with ITS services or with Fog Computing Node (FCN) orchestration. In addiction, they do not have implemented the architecture, but pointed out important future works:

- The creation of a Fog orchestration model that takes into account the capabilities of SDN to support Fog Computing;
- The implementation of an SDN protocol to optimize the reorganization of datapaths, responding to the mobility nature of vehicular networks;
- The improvement of load balancing techniques;
- The need to implement and evaluate the proposed architecture in a simulation environment or on a real environment.

In their work, the authors of [9] present a comprehensive review of Software-Defined Vehicular Ad hoc Networks (SDVN). They envision research challenges for SDVN, among which are the management of topological changes and the adaptive reconfiguration of the network.

The authors in [11] wrote a review of VANET simulators. They compare available simulators in many dimensions, including support for new technologies such as SDN and Edge Computing. However, the study does not mention that existing simulators could address service orchestration in Vehicular Fog Computing Networks using SDN.

There is still a lack of work in the literature that comprehensively evaluates SDVFN scenarios, with the ability to orchestrate ITS services/applications in a Fog Computing paradigm to be consumed by connected vehicles, as well as promoting the network's ability to adapt to the mobility nature of vehicular networks.

Experimental environments are necessary to put IoV, SDN, and Vehicular Fog Networks models into practice. However, it is not always possible to implement real testbeds, either because of the cost of the physical construction of such environments or because of the disruption model of the experiment when compared to usual models.

The simulation of an SDVFN scenario, including the orchestration of ITS applications in Vehicular Fog Computing environments and the exploitation of SDN resources, is therefore essential for carrying out research in this field. Simulations can allow the evaluation of various metrics, including vehicle mobility and network communication, as well as other resources, depending on the focus of the research.

However, reviews in the literature, such as [7, 9, 11], do not point to a network simulator that can deal with an SDVFN scenario directly and comprehensively.

Costa et al. [12] performed a comprehensive systematic review of the current literature on orchestration in fog computing, identifying the key functionalities required for efficient orchestration within fog computing settings. The authors stress the necessity of evaluating service orchestration techniques both in simulation and in real-world test environments.

Sarkohaki et al. [13] provide an overview of the latest and leading techniques for service placement in fog-cloud environments. They organize these techniques into seven primary categories based on optimization strategies: exact solutions, approximated solutions, heuristic and meta-heuristic methods, machine learning approaches, game theory strategies, neural network algorithms, and other miscellaneous methods. The authors highlight the absence of a unified and comprehensive simulation environment as a gap, suggesting that its development is vital to advance research and create more effective methods for service placement.

Pallewatta et al. [14] provide a comprehensive classification of current research on the deployment of microservices within Fog computing settings. The authors highlight the crucial role of effective placement algorithms for microservices. They suggest that scalable placement strategies are crucial to fully leverage the modular and scalable nature of microservices and to tackle the specific challenges inherent in Fog computing frameworks.

This paper presents a simulation framework designed to directly support the implementation and evaluation of service orchestration in software-defined vehicle fog networks (SDVFN) with the following capabilities:

- Implement different strategies for selecting Fog Computing Nodes to compare load balancing algorithms;
- Use the SDN paradigm to provide adaptability to datapaths, given the mobility of vehicles;
- Employ a service-oriented routing rule strategy;
- Anticipate the sending of OpenFlow routing rules for the correct reorganization of datapaths, responding to vehicle movement;
- Responsive Microservice Orchestration in Software-Defined Vehicular Fog Computing Networks;
- Enable Migration of Microservice Execution Context Between Processing Zones in Software-Defined Vehicular Fog Networks Simulation.

In addition, an ITS application use case is also implemented to evaluate the proposed framework.

The rest of this document is structured as follows. Section 2 presents the proposed SDVFN simulation approach to test and evaluate the deployment of ITS applications, the adaptation of the simulation framework, and details of the implementation. Section 3 describes the experimental use case, analyzing its effectiveness and the obtained results. Section 4 presents the conclusions about the work.

2. Proposed SDVFN Simulation Architecture

This study introduces a Simulation Architecture aimed at testing and evaluating methodologies to orchestrate microservices allocation and migration in SDVFN, taking into account vehicular mobility. We validate our method employing a use-case scenario that handles computational microservice tasks from connected vehicle nodes in two ways, using static task allocation and using dynamic mobility responsive task allocation. The Eclipse MOSAIC Framework [15] was utilized to implement several functions associated with SDVFN, thereby creating a simulation environment that allows for:

- the prototyping of ITS applications in SDVFN;
- a Service Placement orchestration to perform optimization on processing computational vehicle tasks;
- a service-oriented routing strategy using the [vehicle,service] tuple information;
- distribution of OpenFlow based forwarding rules to switches, and reorganization the forwarding datapaths;
- the implementation of vehicular handover algorithms.

The proposed SDVFN architecture consists of a VANET within the wireless domain and a Software-Defined Fog Computing Network on the infrastructure side. These components are interconnected by RSUs placed along urban transportation routes, as illustrated in Figure 1.

The SDVFN architecture is made up of three layers. The Infrastructure Layer, the Control Layer, and the Application Layer.

The infrastructure layer encompasses all physical hardware components that enable data processing, storage, and communication across the network. This includes vehicles equipped with computational and communication capabilities RSUs, and fog nodes, which assist in performing data processing and storage tasks close to the network edge.

Each RSU depicted in Figure 1 provides a FCN for performing computational tasks and acts as an access point, connecting wireless VANET nodes to SDVFN, and is also responsible for forwarding packets along the communication path. Moreover, each switch maintains a direct connection to an SDN controller via a dedicated channel to exchange OpenFlow messages.

In the control layer, the Fog Orchestration Server and the SDN controller manage the underlying infrastructure. These two elements can monitor network performance, allocate processing tasks, optimize data flow, and enforce policies to meet network requirements. The SDN controller has a comprehensive view of the network, facilitating centralized governance and dynamic configuration of network resources.

The application layer hosts the various services and applications that depend on the network infrastructure and the support provided by both the Orchestration Server and the SDN Controller.

Eclipse MOSAIC [15] was used to build our SDVFN Simulation Architecture. It is a multi-domain and multiscale simulation framework for testing and developing connected and automated mobility solutions. Eclipse MOSAIC uses High Level Architecture (HLA) to standardize interfaces and employs the concept of federates and ambassadors, coupling individual simulators to

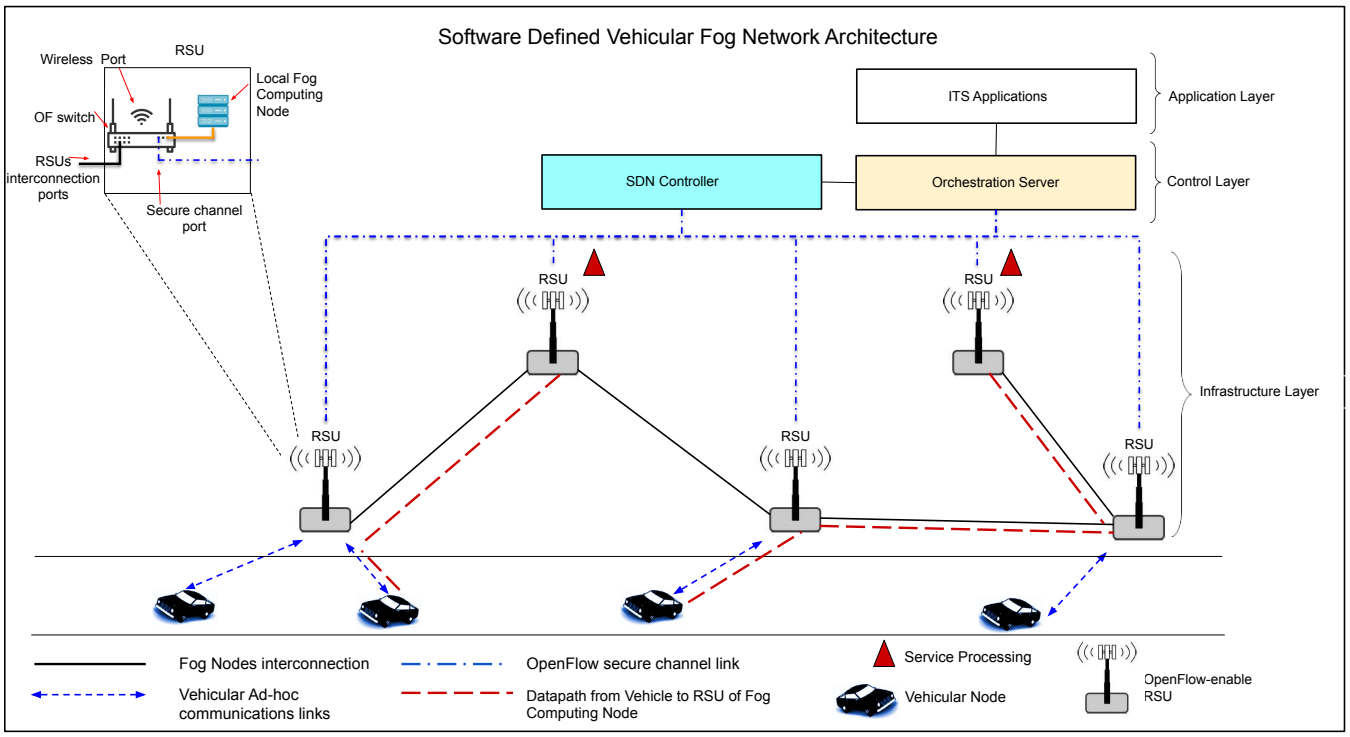

Figure 1. Software-Defined Vehicular Fog Network Architecture.

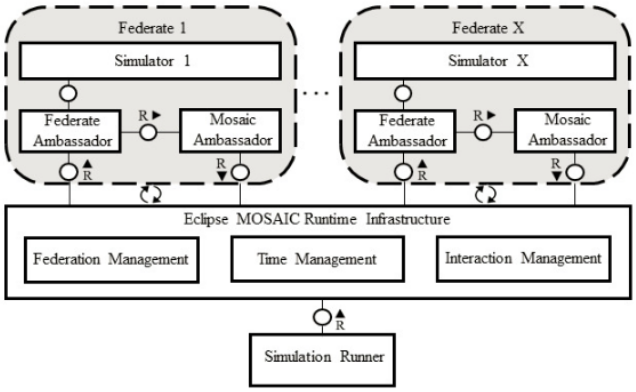

Figure 2. HLA architecture of Eclipse MOSAIC, Available at [16]

the whole environment, thus acting as a co-simulation framework. Federates can be wrapped into a Federate object, which is linked to an Ambassador for direct connection with the MOSAIC runtime infrastructure (RTI), as can be seen in Figure 2. All the management tasks of the simulation life cycle are in charge of the RTI. It includes Federation Management, Time Management, and Interaction Management.

Traffic simulation is carried out using the Eclipse SUMO simulator [17]. All interactions between SUMO and Eclipse MOSAIC are performed through the Traffic Control Interface (TraCI). It is from this interface that it is possible to retrieve information about simulated entities, as well as manipulate their behavior in real-time. On the Eclipse MOSAIC side, the *SumoAmbassador* class interacts with Sumo TraCI and is responsible for implementing the object that represents this Federated Simulator.

The simulation of applications is in charge of the *ApplicationAmbassador* class provided by the built-in MOSAIC application simulator. This simulator prototypes applications in all simulation units, which can be Vehicles, RSU, Traffic Lights, Traffic Management Center (TMC), and Servers.

Applications are deployed into each Unit. They are compiled JAVA classes, which extend the abstract class *AbstractApplication* and implement the *CommunicationApplication* interface.

Each abstract application interacts with the unit's operating system application to gain access to the simulated unit's parameters and perform specific actions of these units, such as navigation and communication between them, using their respective communication modules.

Eclipse MOSAIC offers a domain-specific architecture to couple with a range of simulators: Application simulators, Traffic and Vehicle Behavior Simulators, Network and Communications Simulators, Environment Simulators, and E-Mobility Simulators.

In the field of Network and Communication Simulations, the framework enables the simulation of Cellular Communications and ITS-G5 Ad Hoc Communications. These simulations can be performed using external simulators such as OMNeT++ and NS-3, or using built-in Eclipse MOSAIC simulators. The built-in MOSAIC Simple Network Simulator (SNS) is used to simulate ITS-G5 Ad Hoc networks, while the built-in MOSAIC Cell Simulator is used to simulate cell-network communications. Eclipse MOSAIC does not offer directly wired communications, but we adapt the Cell communications to behave like wired ones.

2.1. SDVFN Infrastructure Layer Simulation

The built-in Eclipse MOSAIC network simulators offer many communication modes. Cellular communications in MOSAIC Cell are available to entities that have a cellular interface enabled. The communication modes are as follows:

- **GeoBroadcast**: Broadcast cellular communication in a specific area.
- **Geocast**: Unicast Cellular Communication in a specific area.
- **Topocast**: Unicast Cellular Communication regardless of geographical area restrictions.

Ad hoc communications simulate the IEEE 802.11p standard in the 5.9 GHz frequency band (5.85-5.925 GHz), which is intended for ITS across seven possible channels. The available modes of ad hoc communications are:

- **GeoBroadcast**: In this mode, messages are broadcasted, and therefore all simulated entities with ad hoc enabled within a specific range will receive the messages.
- **GeoCast**: Ad hoc geocast mode enables unicast communication between two entities on the same ad hoc channel if the receiver's IP address is known. This type of simulated transmission does not impose geographical restrictions on communication between entities.
- **Topobroadcast**: This mode is designed to disseminate information to all nodes within the signal range. This mode can operate in single-hop or multi-hop configurations, and the entities involved in the communication must be operating on the same ad hoc channel. In multi-hop, message dissemination is accomplished through flooding.
- **Topocast**: This mode enables unicast communication in ad hoc networks, subject to no restrictions on signal range and the number of hops.

However, the Eclipse MOSAIC framework does not offer a native method to simulate SDN and Fog Computing Networks and, consequently, SDVFN. As a result, we engineered a suite of Eclipse MOSAIC applications to address this limitation.

Although the coupled OMNET++ and NS-3 simulators can simulate wired networks, Ambassadors do not have this functionality. Furthermore, there is no switch entity available to build the infrastructure network, leading us to implement software switches within the RSU Unit.

OpenFlow Switch Abstraction. The proposed SDVFN architecture simulates a subset of the OpenFlow protocol for communications between the Control Layer and the Infrastructure Layer. In addition, we implemented a novel service-driven packet forwarding strategy that specifies the actions to be taken for each incoming packet.

To establish an infrastructure network which simulates a fog network, we created a packet-switching network interconnected by OpenFlow software switches implemented as a Mosaic Application. Switches are equipped with cellular network modules and ad hoc networks to communicate with other elements of the network. Each switch abstracts four communication ports:

INTRAUNIT_PORT: This port abstracts an interface to provide communication between the OpenFlow switch and the Fog Computing Node of the local RSU where the switch is attached.

ADHOC_PORT: This port uses an abstraction of an ad hoc communication module to provide communication between the OpenFlow switch and the vehicular nodes of SDVFN.

RSUS_PORT: This port uses a cellular communication module to send and receive packets from adjacent switches on the fog side of the SDVFN. It is configured to use Cellular Topocast communication with a constant delay of 10ms, without loss, and without range restriction.

SERVER_PORT: This port abstracts communication with the fog server to exchange SDN OpenFlow messages. Communications are carried out via cellular module using topocast mode and with a constant delay of 20ms without loss and without range restriction.

By reducing communication delays, eliminating range restrictions, and utilizing topocast communications, we enable *RSUS_PORT* and *SERVER_PORT* to function as seamlessly as wired networks.

Each switch contains a flow table comprising various flow rules designed to match incoming data packets and determine the corresponding action. The SDN OpenFlow controller can add, remove, or modify these rules via the connection, using the *SERVER_PORT*.

The infrastructure component of the SDVFN was set up using RSU OpenFlow switches, which provide standard OpenFlow packet forwarding rules. Communications occur through cellular topocast transmissions via the *RSUS_PORT* of each switch.

2.2. SDVFN Control Layer Simulation

The SDVFN control layer consists of the network Orchestration Server and the SDN controller, as can be seen in Figure 1. The Orchestrator coordinates the allocation of services on Fog Computing Nodes and optimizes communication paths to ensure low latency and high throughput.

By integrating SDN with Fog Computing, the SDVFN supports services that are sensitive to location and require a shorter response time. In this context, the Orchestrator communicates with the SDN Controller to manage the routing paths between vehicles and fog nodes.

The Orchestrator maintains a comprehensive data set of vehicular metrics, including positional coordinates, service utilization, velocity, directional heading, and the RSU access point of the vehicle.

The SDN controller constitutes a critical component of the SDVFN. It encompasses a subset of conventional OpenFlow Controller Functions tailored to address the specific requirements of the SDVFN context. The controller processes packet-in messages transmitted from the switches via the secure OpenFlow channel, thereby allowing the addition, removal, or modification of matching rules on the switches.

Additionally, the SDN Controller maintains a network topology model to retain an abstract perspective of the whole Fog Network, encompassing the flow table of every switch. This abstraction is crucial for determining the best paths for packet forwarding and subsequently applying the forwarding rules within the switches. The abstraction of the topology includes only the infrastructure side of the SDVFN, where there are OpenFlow switches.

2.3. SDVFN Control Layer Modules

The SDVFN is made up of several elements. Figure 3 shows how the main simulation components are distributed. The modules of both the SDN Controller and the Fog Orchestration Server play a pivotal role in the SDVFN orchestration.

Fog Orchestrator Server Modules. The *Microservice Registration Module* is responsible for recording relevant information about each microservice in the SDVFN, with the following objectives:

- Register the replication strategy used for each microservice in the SDVFN, including the maximum total number of replicas for each microservice.
- Maintain a list of fog computing nodes where the replicas of each microservice are located.
- Maintain a registration of all vehicle customers for each microservice.

The *Mobility Monitoring Module* monitors the information about the vehicle as it moves. Periodically, registered vehicles send information about their speed, heading, and geographical position to the Fog Orchestrator Server, including which RSU is used by each vehicle to connect to the SDVFN. In addition, when a vehicle is about to perform a handover from the actual RSU to another, it sends a handover notification to the Fog Orchestrator Server. Subsequently, the *Mobility Monitoring Module* requests the SDN Controller to anticipate the distribution of the OpenFlow rules to the new datapath from the next RSU to the Fog Computing Node. The actual and next path information is maintained by the *Mobility Monitoring Module* for each pair of [vehicle,service].

The *MicroService Placement Optimization Module* is responsible for implementing the algorithms in charge of the allocation and reallocation of microservice replicas according to the replication strategy defined in *Microservice Registration Module* for each specific Microservice. This means that each microservice can be submitted to different service placement models. This can be illustrated by deploying Kubernetes pods on FCNs.

With the movement of vehicles and the consequent handover from one RSU to another, the initial geographical positioning of microservice replicas may have become unsuitable to optimize communication latency between vehicles and microservices. When this happens, the *Microservice Placement Optimization Module* can decide to rearrange the distribution of the replicas of a given microservice, following their respective optimization strategies. The module calculates the optimal distribution of replicas for the current scenario, and replicas are migrated to the designated FCNs.

Each migration generates the need to reorganize the datapaths between FCNs and the RSUs for all vehicles that consume the same service that is being migrated. The Fog Orchestrator Server asks the current FCN to migrate the replica to the new FCN that will host the replica and, after the migration, asks the SDN Controller to trace the new datapaths and distribute the OpenFlow forwarding rules on the switches affected by the change.

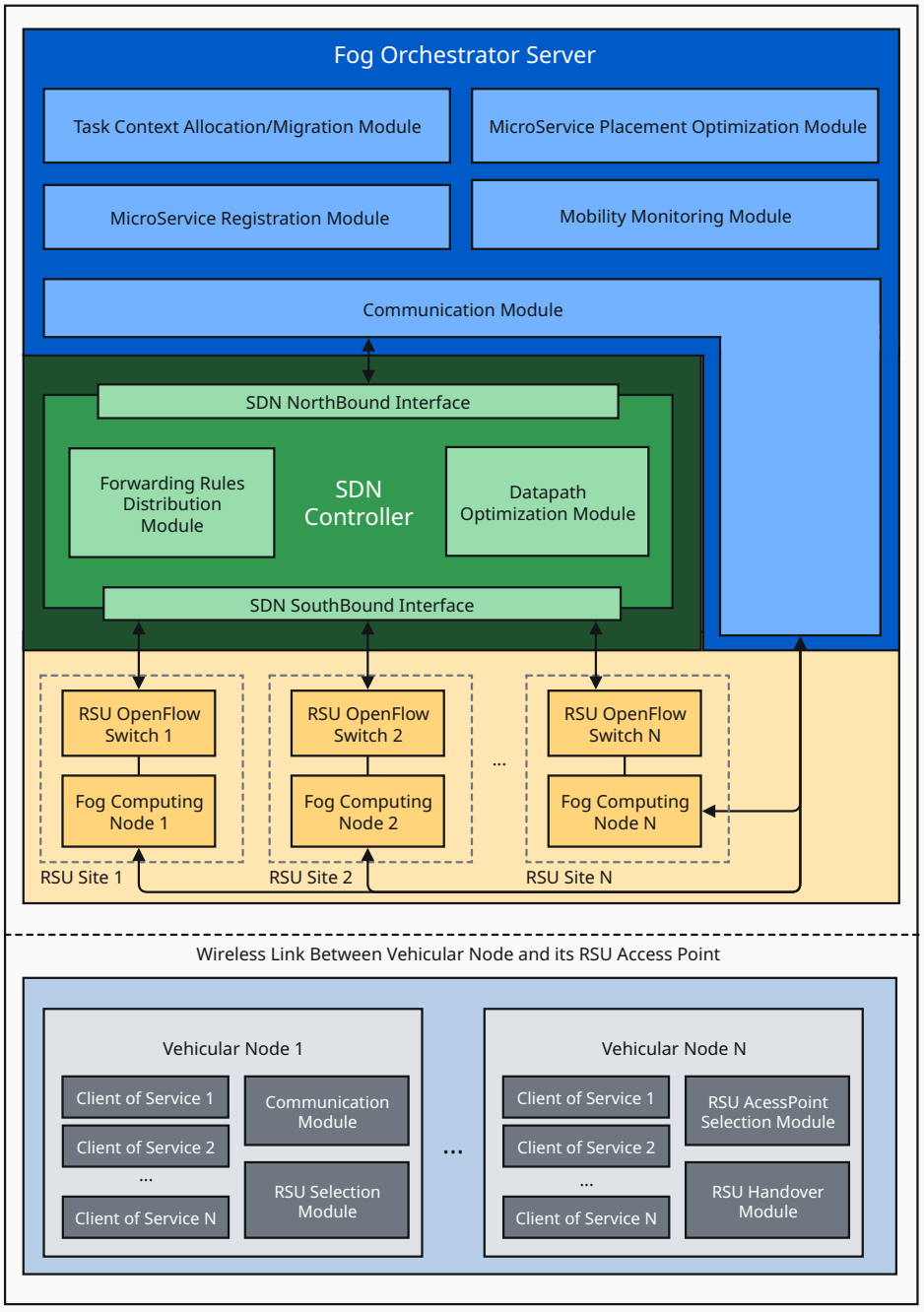

Figure 3. Component Vision of SDVFN Architecture.

The *Task Context Allocation/Migration Module* is responsible for selecting the appropriate microservice replica (i.e., Kubernetes pods) to process vehicle requests. The *Service Context Allocation/Migration Module* can reallocate microservice replicas applying some strategy. In addition, the module can migrate the execution contexts between replicas of the same microservice. In this case, the replica remains in the same FCN, but the vehicle is now served by another FCN that has another replica of the same microservice.

The execution context migration reallocates the current state of the task that is being processed, independently of any replica reallocation. This means that execution contexts can be transferred from one microservice replica to another, based on any strategy designed to respond to vehicle movement. As a result, this migration ensures the balance of load between replicas, supporting the achievement of QoS restrictions, such as selecting one that is not overloaded and geographically closer to the vehicle.

SDN Controller Modules. The *Datapath Optimization Module* depicted in Figure 3 generates a topological representation of a Fog Network and optimizes the network's data pathways. Currently, it uses the Dijkstra algorithm, which is a popular algorithm for finding the shortest path between nodes in a graph. However, the system is designed to be flexible, allowing additional algorithms to be implemented in the future if necessary.

Forwarding Rules Distribution Module registers datapaths used by vehicles to access specific microservices. The referred module contains crucial information on how data are routed through the network. Thus, when the *Datapath Optimization Module* determines a new datapath, the *Forwarding Rules Distribution Module* compares this new path with the existing one in its registers.

The module not only distributes the new routing rules to the network switches in the new section of the datapath, but also cleans up outdated rules from switches that were in the section of the datapath that are no longer needed in the current configuration.

These modules help the SDN Controller to optimize network resources and ensure optimal routing for vehicles accessing microservices, by dynamically updating and removing routing rules as needed.

OpenFlow Service-Driven Forwarding Rules on SDVFN. The proposed SDVFN Framework uses personalized rules for each microservice execution context of a specific vehicle. This means that when a vehicle uses various services, the microservice execution context of each vehicle can be handled by different nodes within the SDVFN. By doing this, the network can effectively manage traffic by assigning a unique rule for each combination of vehicle and microservice, allowing efficient and flexible data routing and resource allocation in SDVFN.

When a vehicle sends the first message to the SDVFN, the RSU access point sends the vehicle's data to the Orchestrator and awaits a decision on which Fog Computing node will process the requested microservice.

The Orchestrator, in turn, records vehicle information and chooses one of the microservice replicas in the respective FCN to attend the vehicle and requests to the SDN Controller to calculate the best datapath between the vehicle and the selected FCN to host the execution context of the [vehicle,service] pair. Then, it sends the customized OpenFlow [vehicle,service] forwarding rules to all switches along the datapath. Vehicles sends its data periodically to the SDVFN to help Orchestrator and SDN controller on network mobility responsiveness.

3. Experimental Use Case Analysis and Results

To validate our work, we developed an experimental use case scenario to evaluate the suitability of the proposed SDVFN architecture to orchestrate microservices, including the migration of its execution contexts responding to vehicle mobility. In addition, we analyze its ability to apply various strategies to optimize data traffic in the SDVFN. To achieve the research objectives, we use the following requirements:

1. The SDVFN should allow for the selection of a specific replica of a given microservice allocated to a FCN (Fog Computing Node) to process requests sent by a given vehicle, thus evaluating the capacity of the proposed architecture to test load balancing techniques on distribute replicas of a microservice.

2. The simulation environment must be able to establish an optimized datapath for each vehicle from its RSU access point to its respective processing Fog Node within the SDVFN.

3. It must be able to effectively use the SDN paradigm to adapt to vehicular mobility, by rearranging data forwarding paths along with vehicle handovers.

4. It must implement a seamless handover between the RSUs. In this case, the main goal is to assess how effectively the environment can facilitate various handover methodologies.

5. It must have the capability to migrate the microservice execution context from one microservice replica to another, responding to vehicular mobility.

6. The SDVFN must allow algorithms to optimize the use of customized strategies in migrating microservice execution contexts.

The experimental environment was created using Eclipse SUMO for the mobility simulation and the Eclipse MOSAIC Multi-Domain Framework for the implementation of network communications, and the algorithms that implement the SDN OpenFlow protocol and its modules, Orchestrator and its modules, and the applications in the switches, FCNs, and vehicles. The experiments are geographically contextualized using a partial map of the city of Rio Verde, located in the Brazilian state of Goiás, as shown in Figure 4. The SDVFN is made up of 58 RSUs interconnected by OpenFlow switches.

Each Roadside Unit (RSU) signal covers a radius of 200 meters. They are strategically positioned to provide optimal average signal coverage. Vehicles remain within the signal range of at least 2 RSUs, but some zones are

Figure 4. Geographic distribution and network communication topology of the RSUs in the SDVFN experimental scenario.

intentionally covered by just 1 RSU or outside of any RSU signal coverage.

Communications between vehicles and the RSU switches consistently employ wireless ad hoc connections, incurring a 10 ms latency for each direction of network packets. Similarly, it takes 20 ms for communications between RSUs OpenFlow switches. Furthermore, interaction times between OpenFlow switches and the SDVFN servers, namely the Orchestrator and the SDN Controller, are 50 ms in each direction. The simulation also accounts for a 3 ms communication time between the OpenFlow switch and the FCN at each RSU location. This arrangement is quite the same as our previous work [18] and can be parametrized in the framework.

We used two comparative experimental scenarios: The first is the same as used in [18], with seamless handovers, and the second implements a microservice context migration in microservice orchestration in response to vehicular mobility. Both scenarios are in the context of a limited number of microservice replicas.

3.1. Comparative experimental scenarios

The Static Task Context Allocation Scenario. The Baseline Scenario is the Static Task Allocation Scenario. It uses ten Fog Computing Nodes that handle tasks transmitted by vehicles, all providing the same microservice. In Figure 4, these nodes are represented as red RSUs and are identified with the following numbers: RSUs 0, 6, 13, 14, 22, 28, 34, 43, 48, 54.

In the baseline scenario, when a vehicle connects to the SDVFN for the first time, the orchestrator applies the resource allocation strategy and defines which FCN will process the vehicle's requests. In this scenario, the vehicle's tasks are always processed in the same FCN throughout the simulation.

The Orchestrator applies the round-robin algorithm to distribute the processing tasks of each vehicle to the Microservice replica in one of the ten Fog Computing Nodes on the SDVFN. In the SDVFN architecture, this straightforward load balancing can be replaced by other alternative methods.

The orchestrator makes the decision regarding the allocation of processing tasks, in accordance with the round-robin algorithm to select an FCN for processing tasks. In turn, the SDN Controller defines an optimized datapath to forward data between the OpenFlow switch in the RSU Access-Point (RSU-AP) and the previously selected FCN which has a microservice replica.

Even in the baseline scenario, the SDVFN implements two mechanisms for adapting to vehicle mobility: The first is the ability to reorganize the datapath, reacting to changes in the vehicle's connection points with the SDVFN. In other words, whenever the vehicle connects to another RSU of the SDVFN, the data paths will be reorganized to obtain the best path to the data, as presented in our previous work [19]. The second mechanism for the mobility response is to anticipate both the calculation of the next datapath and the distribution of the OpenFlow rules in advance throughout the next datapath that will be used when the vehicle performs handover to the next RSU-AP, reducing the increase in communication latencies and consequently promotes seamless handovers, as presented in [18]. The new rules are distributed to the OpenFlow switches in new sections that will be part of the next optimized datapath that connects the next RSU-AP and the FCN. To get a minimum cost to distribute the rules, the sections shared between the current datapath and the next datapath do not need to receive new rules.

Although the baseline achieves excellent adaptation to the mobility scenario, it can still be noted from the graph in Figures 6, 7, 8, 9, when the SDVFN is using the "Static Task Allocation", and the vehicle is changing its RSU-AP, it causes a significant variation in the number of hops in the datapaths and consequently a big variation in latency, increasing not only the average latency but also the jitter in the communication between vehicles and the microservice.

The Dynamic "Cluster Head" Task Context Allocation Scenario. The second experimental scenario also uses an approach with the same limited number of 10 microservice replicas and has mobility response mechanisms implemented for the baseline. However, in this second scenario, each replica of a given microservice is allocated to a FCN called the "Cluster Head" of the microservice in that zone. This "Cluster Head" is responsible for processing requests from all

vehicles that use as RSU-AP any of the members of that cluster zone, as shown in Figure 5.

Figure 5. Zones Distribution for Dynamic "Cluster Head" Microservice Task Context Allocation in SDVFN.

In this proposed scenario, the definition of the processing zones and the composition of their members is performed once based on the parameters defined for the simulated microservice. The Orchestrator Server performs the service placement of each microservice replica with the aim that each processing zone tends to have the same maximum number of hops between its most distant members (in hops) and the FCN "Cluster Head" of the Microservice in that zone. Each microservice has an abstract division of the SDVFN area into processing zones, based on the number of replicas available for each microservice.

In this proposed scenario, as the vehicle moves and performs the handover to another RSU-AP, the datapaths are rearranged as in the baseline scenario. However, when a vehicle moves from RSU-AP to an RSU that is part of another processing zone, the Orchestrator Server detects this situation and migrates the execution context of vehicle requests to the "Cluster Head" FCN of the next processing zone. This migration also triggers the rearrangement of the datapaths between the Vehicle's RSU-AP and the new FCN that will handle the vehicle's requests in this new processing zone.

3.2. Analysis of Results

The graphs in Figures 6, 7, 8 and 9 show the communication latencies between vehicles 1, 2, 3, and 4 and their respective FCN runners throughout the simulation time. The minimum latency is 26 ms when the vehicle is directly connected to the RSU where the microservice FCN runner is allocated. In this case, we consider the vehicle to be at a distance of 0 hop from the microservice execution context. The latency increases by 40 ms for each additional hop that is added as a result of handover. Depending on the vehicle's route, there may be increments greater than 40 ms because the vehicle routes do not coincide with the data paths, and the handover methodology used prioritizes minimizing unnecessary handovers, as presented in [18].

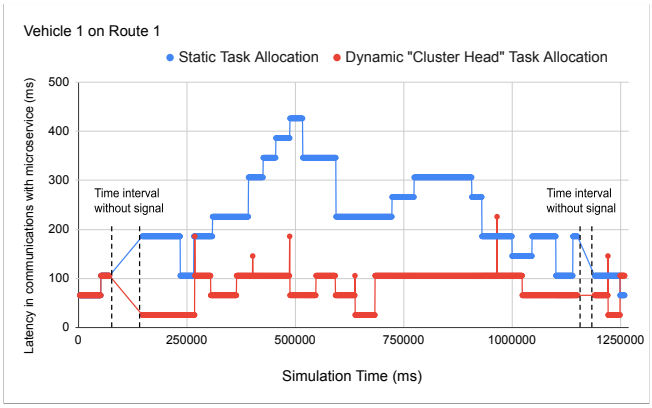

Figure 6. Vehicle 1 communications latencies.

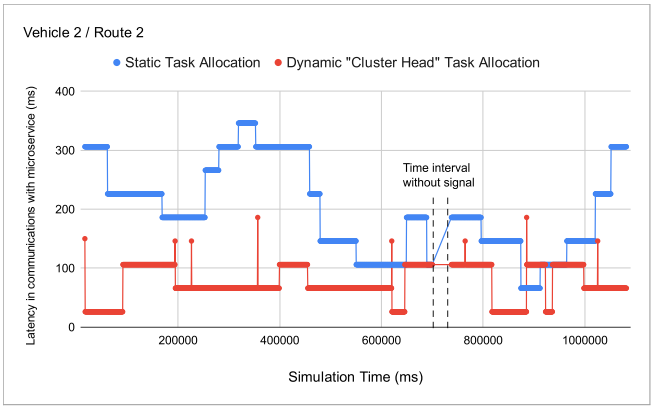

Figure 7. Vehicle 2 communications latencies.

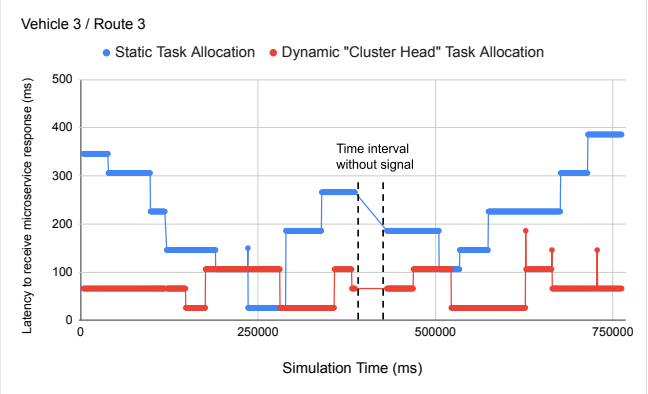

Figure 8. Vehicle 3 communications latencies.

Table 1. Experimental results using Static Task Allocation Strategy versus Dynamic "Cluster Head" Allocation Strategy in SDVFN.

	Vehicle 1		Vehicle 2		Vehicle 3		Vehicle 4	
	Static Task Allocation	Dynamic "Cluster Head" Task Allocation	Static Task Allocation	Dynamic "Cluster Head" Task Allocation	Static Task Allocation	Dynamic "Cluster Head" Task Allocation	Static Task Allocation	Dynamic "Cluster Head" Task Allocation
Min. Latency	66 ms	26 ms	66 ms	26 ms	26 ms	26 ms	106 ms	26 ms
Max. Latency	426 ms	226 ms	346 ms	186 ms	386 ms	186 ms	346 ms	186 ms
Avg. Latency	210.16 ms	78.88 ms	189.15 ms	77.89 ms	192.49 ms	69.45 ms	194.66 ms	78.39 ms
Min. Hops	1	0	1	0	0	0	2	0
Max. Hops	10	5	8	4	9	4	8	4
Avg. Hops	4.6	1.32	4.07	1.29	4.16	1.08	4.21	1.31
Jitter	71.79 ms	28.69 ms	62.96 ms	24.91 ms	75.95 ms	23.37 ms	46.42 ms	24.30 ms
Total of Messages	2102	2766	1968	2502	1355	1771	1961	2521

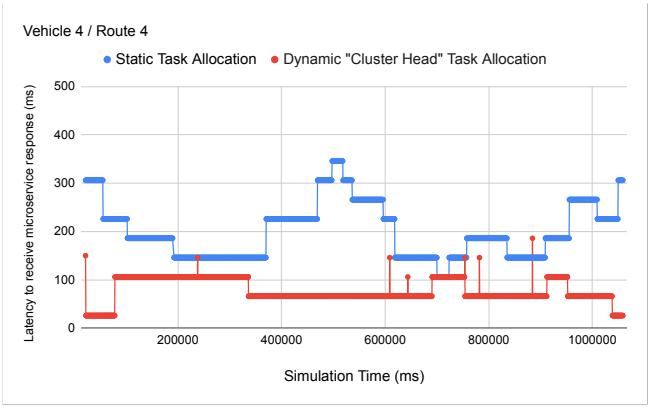

Figure 9. Vehicle 4 communications latencies.

In these graphs, it can be seen that when SDVFN is using Static Task Allocation, in blue, as the vehicle moves along its route and performs handovers, there is a change in the number of hops to reach the FCN runner and, consequently, it causes a large variation in the latency of communications between the vehicle and the microservice. On the other hand, when analyzing the results for Dynamic Task Allocation using the "Cluster Head" strategy, shown in red, it confirms what was expected from the distribution seen on the map in Figure 5. That is, when traveling along the route, the number of hops between the vehicle and the FCN runner does not increase in the same way as it does in the static task allocation.

When connecting to an RSU-AP located in another processing zone, the number of hops would increase, but with the migration of the execution context to this new processing area, both the number of hops and the latency remain lower than in the static approach. However, even in the approach with context migration, there may be some sparse messages with higher latency. It occurs when there is a communication in progress at the same time as the task migration, but with a minimum impact when compared with the total number of messages. As an example, in Figure 6 it is possible to see that only 6 messages or 0.216% of all 2,776 messages had this effect.

In Table 1, the experimental results are presented comparing the static task allocation strategy with a dynamic "Cluster Head" allocation strategy in SDVFN. It is possible to see better results for the proposed methodology in all metrics.

By analyzing the results shown in Table 1, we can conclude that the average reduction in communication latency for vehicles 1, 2, 3, and 4 was 62.47%, 58.82%, 63.92%, and 59.73%, respectively. Thus, we can see that the overall average latency was reduced by 61.10%, which represents a significant decrease when we move from a static allocation scenario to a scenario that applies the migration of microservice execution contexts following the dynamic allocation methodology with "Cluster Heads".

As can be seen in Table 1, the results also show that the number of hops required for vehicles to reach the microservice replica drops from an overall average of 4.21 hops in the static allocation scenario to 1.30 hops when the proposed strategy of "Cluster Heads" is applied, which means 69.12% of reduction.

In the same way, the overall jitter metric decreased from 67.35 ms in static allocation scenario to 24.60 ms in the "Cluster Head" scenario. which means 63.48% of reduction, offering communication with fewer latency variations.

The last row of the table shows the difference in the number of messages exchanged between the vehicle and the microservice when applying the static allocation strategy and also when applying the 'Cluster Head' strategy. As can be seen, there is an increase in the transmission rate of 31.59%, 27.13%, 30.70% and 28.56% for vehicles 1, 2, 3 and 4, respectively, indicating faster responses, in line with the other results.

It is important to note that in a static allocation scenario, the distance in hops of communications varies based on a direct relation between the positions of the RSUs used by the vehicle on its route and the geographical position of the microservice replica, considering the entire geographical coverage and topology of the SDVFN network. However, when we use the processing zone strategy with "Cluster Heads", we confine this same effect to only the geographic area of the processing zone itself. Therefore, the more replicas of a Microservice are distributed in the SDVFN, the smaller the processing zones will be, and thus it will be possible to have fewer hops and consequently lower latencies, lower jitters, and higher transmission rates in communications.

4. Conclusion

SDN networks have shown great potential in the implementation of ITS applications in IoV scenarios, especially in the implementation of SDVFN. The proposed SDVFN simulation framework makes possible the investigation of various strategies for selecting Fog Computing nodes, including load balance algorithms. SDN networks have shown significant promise for deploying ITS applications within IoV contexts, particularly in the realm of establishing SDVFN. The SDVFN simulation framework that has been proposed enables exploration of multiple strategies for choosing Fog Computing nodes, such as load balancing algorithms.

The Simulation framework successfully fulfilled its objectives by developing adaptive datapaths aligned with vehicle movements using the SDN OpenFlow capability. The environment is specifically designed to allow for detailed comparative analysis of different methodologies of service orchestration in SDVFN.

The use case demonstrates that Orchestrator, together with SDN Controller, is capable of promoting service placement of microservice replicas, rearranging data paths between RSU-AP and the FCN running the microservice, responding to mobility. The platform can promote smooth handover strategies and increase network efficiency by applying the migration of execution contexts between microservice replicas that are closer and better suited to serve the vehicle. It enables optimal workload distribution across fog nodes, simultaneously decreasing infrastructure network traffic and reducing communication delays.

The anticipation of forwarding rules demonstrated effectiveness, primarily through utilizing the OpenFlow rules based on [vehicle, service] pair. These rules are uniquely assigned to each vehicle, ensuring that they do not interfere with the forwarding of other packets and allowing different routing to different microservices, even to the same vehicle. In the same way, it enables different vehicle task processing allocation on different FCN, even if they are consuming the same microservice.

It also correctly implements the adaptability of datapaths through the SDN paradigm, even with vehicular mobility, and proves to be efficient in anticipating route sending, based on decision making generated by algorithms that predict the next RSU Access Point for the vehicle.

The next steps include migrating the microservice replicas - since the current implementation only migrates the execution context - and improving support for simulating multiple microservices in parallel. We intend to implement dynamic strategies to define processing zones and their respective "Cluster Head". The framework could also respond dynamically to the workload of the FCNs and the distribution of vehicles in each geographic region of the SDVFN.

In addition, we will expand the range of metrics to evaluate the system's performance, including security features, and also make the framework available for free use by the scientific community.

5. Acknowledgements

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Unit Project Scope UID/00319/Centro ALGORITMI (ALGORITMI/UM)

References

[1] Yuan, T., Da Rocha Neto, W., Rothenberg, C.E., Obraczka, K., Barakat, C. and Turletti, T. (2022) Machine learning for next-generation intelligent transportation systems: A survey. *Transactions on Emerging Telecommunications Technologies* **33**(4): 1–35. doi:10.1002/ett.4427.

[2] Darwish, T.S. and Abu Bakar, K. (2018) Fog Based Intelligent Transportation Big Data Analytics in The Internet of Vehicles Environment: Motivations, Architecture, Challenges, and Critical Issues. *IEEE Access* **6**: 15679–15701. doi:10.1109/ACCESS.2018.2815989.

[3] Sharma, S. and Kaushik, B. (2019) A survey on internet of vehicles: Applications, security issues & solutions. *Vehicular Communications* **20**: 100182. doi:10.1016/j.vehcom.2019.100182, URL

https://linkinghub.elsevier.com/retrieve/pii/S2214209619302293.

[4] Langley, D.J., van Doorn, J., Ng, I.C., Stieglitz, S., Lazovik, A. and Boonstra, A. (2021) The Internet of Everything: Smart things and their impact on business models. *Journal of Business Research* **122**(January 2020): 853–863. doi:10.1016/j.jbusres.2019.12.035, URL https://doi.org/10.1016/j.jbusres.2019.12.035.

[5] Farias da Costa, V.C., Oliveira, L. and de Souza, J. (2021) Internet of Everything (IoE) Taxonomies: A Survey and a Novel Knowledge-Based Taxonomy. *Sensors* **21**(2): 568. doi:10.3390/s21020568, URL https://www.mdpi.com/1424-8220/21/2/568.

[6] Ji, B., Zhang, X., Mumtaz, S., Han, C., Li, C., Wen, H. and Wang, D. (2020) Survey on the Internet of Vehicles: Network Architectures and Applications. *IEEE Communications Standards Magazine* **4**(1): 34–41. doi:10.1109/MCOMSTD.001.1900053, URL https://ieeexplore.ieee.org/document/9088328/.

[7] Truong, N.B., Lee, G.M. and Ghamri-Doudane, Y. (2015) Software defined networking-based vehicular Adhoc Network with Fog Computing. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015* : 1202–1207doi:10.1109/INM.2015.7140467.

[8] Liu, K., Xu, X., Chen, M., Liu, B., Wu, L. and Lee, V.C. (2019) A Hierarchical architecture for the future internet of vehicles. *IEEE Communications Magazine* **57**(7): 41–47. doi:10.1109/MCOM.2019.1800772.

[9] Bhatia, J., Modi, Y., Tanwar, S. and Bhavsar, M. (2019) Software defined vehicular networks: A comprehensive review. *International Journal of Communication Systems* **32**(12): e4005. doi:10.1002/dac.4005, URL http://doi.wiley.com/10.1002/dac.4005.

[10] Ku, I., Lu, Y., Gerla, M., Gomes, R.L., Ongaro, F. and Cerqueira, E. (2014) Towards software-defined VANET: Architecture and services. In *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)* (IEEE): 103–110. doi:10.1109/MedHocNet.2014.6849111, URL http://ieeexplore.ieee.org/document/6849111/.

[11] Weber, J.S., Neves, M. and Ferreto, T. (2021) VANET simulators: an updated review. *Journal of the Brazilian Computer Society* **27**(1): 8. doi:10.1186/s13173-021-00113-x, URL https://journal-bcs.springeropen.com/articles/10.1186/s13173-021-00113-x.

[12] Costa, B., Bachiega, J., Rebouças, L., Carvalho, D.E., Araujo, A.P.F. and Rebouças De Carvalho, L. (2022) Orchestration in Fog Computing: A Comprehensive Survey. *ACM Comput. Surv* **55**. doi:10.1145/3486221, URL https://doi.org/10.1145/3486221.

[13] Sarkohaki, F. and Sharifi, M. (2024) Service placement in fog–cloud computing environments: a comprehensive literature review. *The Journal of Supercomputing* **80**(12): 17790–17822. doi:10.1007/s11227-024-06151-4, URL https://link.springer.com/10.1007/s11227-024-06151-4.

[14] Pallewatta, S., Kostakos, V. and Buyya, R. (2023) Placement of Microservices-based IoT Applications in Fog Computing: A Taxonomy and Future Directions. *ACM Computing Surveys* **55**(14s): 1–43. doi:10.1145/3592598.

[15] Schrab, K., Neubauer, M., Protzmann, R., Radusch, I., Manganiaris, S., Lytrivis, P. and Amditis, A.J. (2023) Modeling an ITS Management Solution for Mixed Highway Traffic With Eclipse MOSAIC. *IEEE Transactions on Intelligent Transportation Systems* **24**(6): 6575–6585. doi:10.1109/TITS.2022.3204174.

[16] Eclipse Mosaic Website (2024), Eclipse Mosaic: A Multi-Domain and Multi-Scale Simulation Framework for Connected and Automated Mobility. URL https://eclipse.dev/mosaic/.

[17] Lopez, P.A., Wiessner, E., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y.P., Hilbrich, R. *et al.* (2018) Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (IEEE): 2575–2582. doi:10.1109/ITSC.2018.8569938, URL https://ieeexplore.ieee.org/document/8569938/.

[18] Alvarenga, L.D.C., Sousa, P. and Costa, A. (2024) Seamless Handovers in Software-Defined Vehicular Fog Networks. In *2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS)* (IEEE): 1–8. doi:10.1109/ISAS64331.2024.10845537.

[19] Alvarenga, L.D.C., Sousa, P. and Costa, A. (2025) IoV Simulation Architecture for Software-Defined Vehicular Fog Network Orchestration. 40–54. doi:10.1007/978-3-031-84426-3_4, URL https://link.springer.com/10.1007/978-3-031-84426-3_4.