5G-Slicing-Enabled Scalable SDN Core Network: Toward an Ultra-Low Latency of Autonomous Driving Service

Djabir Abdeldjalil Chekired, *Student Member, IEEE*, Mohammed Amine Togou, *Member, IEEE*, Lyes Khoukhi, *Member, IEEE*, and Adlen Ksentini, *Member, IEEE*

***Abstract*— 5G networks are anticipated to support a plethora of innovative and promising network services. These services have heterogeneous performance requirements (e.g., high-rate traffic, low latency, and high reliability). To meet them, 5G networks are entailed to endorse flexibility that can be fulfilled through the deployment of new emerging technologies, mainly software-defined networking (SDN), network functions virtualization (NFV), and network slicing. In this paper, we focus on an interesting automotive vertical use case: autonomous vehicles. Our aim is to enhance the quality of service of autonomous driving application. To this end, we design a framework that uses the aforementioned technologies to enhance the quality of service of the autonomous driving application. The framework is made of 1) a distributed and scalable SDN core network architecture that deploys fog, edge and cloud computing technologies; 2) a network slicing function that maps autonomous driving functionalities into service slices; and 3) a network and service slicing system model that promotes a four-layer logical architecture to improve the transmission efficiency and satisfy the low latency constraint. In addition, we present a theoretical analysis of the propagation delay and the handling latency based on GI/M/1 queuing system. Simulation results show that our framework meets the low-latency requirement of the autonomous driving application as it incurs low propagation delay and handling latency for autonomous driving traffic compared to best-effort traffic.**

***Index Terms*— 5G networks, autonomous driving, network functions virtualization, network slicing, software-defined networking.**

I. Introduction

THE emerging 5G technology is foreseen as the promising solution to improve network performance and management while ensuring high data rate and enhanced quality of service. It is expected to support a variety of vertical industries such as manufacturing, automotive, healthcare, energy, media and entertainment [1]. Such verticals will spark different use cases, imposing new set of requirements (e.g., scalability, latency, reliability and availability) that the currently deployed networks, which are based on the "one-fit-all" conceptual paradigm, are not able to meet [2], [3]. Accommodating the new applications while supporting existing services imply having a programmable and flexible network infrastructure that can be shared by different network technologies (e.g., IoT, cellular and vehicular).

One way to achieve such a design is a concept called network slicing. It consists of splitting the physical network infrastructure into multiple logical networks, referred to as slices. These slices are controlled and managed independently by the slice owners, which can be of two types: Over-The-Top (OTT) service providers and Virtual Mobile Network Operators (VMNO) [4]. Each slice is allocated a set of network functionalities that are selected from the shared network infrastructure. These functionalities can be virtualized using technologies such as SDN and NFV. Indeed, an SDN-based architecture was proposed in [5] to enable efficient and scalable network slicing. It deploys a controller between slice owners and the shared network infrastructure that provides an abstract view of the network resources to the slice owners while enabling them to transmit their service requirements to the network infrastructure using northbound interfaces.

On the other hand, according to data from the Boston Consulting Group [6], the market for autonomous vehicles (AVs) will grow to 42 billion USD by 2025. It is also expected that by 2020, 10 million autonomous driving cars will be on the road while there will be more than 250 million smart vehicles connected to wireless access networks. However, there are still policy and safety considerations that must be addressed before autonomous driving vehicles travel our highways and small town roads, right now the biggest constraint to autonomous adoption is technical, and network latency is the top technical concern.

Like standard vehicles where a defective system may activate a notification sensor on the dashboard, autonomous vehicles generate and transmit data and information messages [7]. But the information that they generate and transmit doesn't just stay within the vehicles. AVs are in constant communication with networks and maybe even with other nearest vehicles. Intel estimates that once

Manuscript received February 15, 2019; revised March 13, 2019 and June 18, 2019; accepted June 19, 2019. Date of publication July 10, 2019; date of current version August 6, 2019. *(Corresponding author: Djabir Abdeldjalil Chekired.)*

D. A. Chekired and L. Khoukhi are with the Autonomic Networking Environment, Charles Delaunay Institute, University of Technology of Troyes, 10004 Troyes, France (e-mail: djabir_abdeldjalil.chekired@utt.fr; lyes.khoukhi@utt.fr).

M. A. Togou is with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin, D09 E8H4 Ireland (e-mail: mohammedamine.togou@dcu.ie).

A. Ksentini is with the Communication Systems Department, EURECOM, 06410 Sophia-Antipolis, France (e-mail: adlen.ksentini@eurecom.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2019.2927065

adopted, autonomous driving vehicle may generate more than 4 terabytes of data per day [8]. However, the current communication networks do not have the bandwidth or the latency to support such a reality. Thus, the movement to make autonomous driving vehicles an everyday presence on the roadways must first realize an essential transformation on the current communication and network technologies. The emerging 5G technology can be the good solution, it has this potential: an increased number of connections allowed at any one time (100 billion); low latency (1 ms, compared to current 4G 50ms); and high throughput (10 Gbps) [9], [10]. However, latency isn't the only consideration; high reliability is just as very important for autonomous driving. To avoid overloading in the core networks, recent research study will likely have to trust on hybrid networks that combine a variety of methods, including SDN, edge computing, and fog computing.

In this paper, our goal is to provide a system architecture that combines network slicing with SDN and NFV technologies to enhance the quality of service of the autonomous driving application. For this system to be efficient, two key challenges are to be addressed: end-to-end reliability and ultra-low latency. While the former deals with resource provisioning and service availability, the latter tackles the issue of transmitting high-rate traffic between servers and autonomous vehicles through the core network. These challenges are very complex given the distinctive mobility patterns of vehicles as well as the abrupt changes in channel conditions. As a result, a solution would require adequate orchestration of network functionalities between all the technologies deployed in order to meet the requirements of the autonomous driving application.

Our contribution is fourfold. First, we design a distributed and scalable SDN core network architecture based on fog, edge and cloud computing. It uses a clustering technique to partition the network into multiple fog cells, each of which is managed by a controller. This is to balance the traffic load among different controllers, reducing therefore network congestion and message transmission overhead while guaranteeing appropriate connectivity and service continuity. Second, we design and implement a network slicing function that maps autonomous driving functionalities (i.e., localization, perception, planning and system management) into service slices to ensure efficient access to these services. Third, we design a network and service slicing system model to improve the transmission efficiency and satisfy the low latency constraint. In this model, each base station has four logical layers: control, resource management, virtualization and interface. The control layer is responsible for allocating resource blocks to the different slices based on their service requirements. The virtualization layer organizes the different service slices and enables resource blocks sharing among adjacent base stations. The resource management layer allocates the shared resource blocks to the service slices in case all resource blocks at the current base station are used while the interface layer provides an interface for the various predefined autonomous driving functions. Finally, we present an ultra-low latency mathematical framework where we used a GI/M/1 queuing system to define the handling and tail latencies.

The rest of this paper is organized as follows. Section II surveys some related work. Section III describes the scalable SDN core network. Section IV introduces the autonomous driving service and function slicing. Section V describes the network service and the slicing model while Section VI presents the ultra-low latency mathematical framework. Section VII presents the autonomous driving resource management slicing algorithm. Simulation settings and results are presented in Section VIII. Section IX concludes the paper.

II. Related Work

A. Network Slice Architectures Based SDN

There exists various network slicing approaches that have been proposed in the literature. For example, Costanzo et al. [11] proposed a network slicing solution that makes use of an efficient scheduling algorithm, based on the centralized SDN architecture presented in [5], to ensure a real-time allocation of bandwidth to different slices considering their requirements in cloud radio access network. Velasco *et al.* [12] proposed an architecture to enable autonomic slice networking. It has two domains: metro and core, each of which includes: i) a provisioning and reconfiguration module based on SDN; ii) a data analytics module that collects data records from nodes and analyze them; and iii) a slice manager that exports virtualized network resources in the form of network slices through a northbound interface. Unlike both approaches in [11] and [12] which are based on a centralized SDN architecture, our work proposes a new core network based on decentralized and scalable SDN architecture.

Ma *et al.* [13] proposed a management architecture for 5G service-based core network based on SDN and NFV to provide distributed and on-demand deployment of network functions, service guaranteed network slicing, flexible orchestration of network functions and optimal workload allocation. It has two layers: i) service management which is responsible for providing services such as authentication, orchestration and security; and ii) the infrastructure management layer that contains two different SDN controllers: the core SDN which is responsible for deployment and management of network functions, and the flow SDN controller which handles efficient traffic dispatch of the core network. In this way, the core network functions can be deployed in a distributed fashion, easing the deployment of mobile edge computing technology to alleviate the workload on the core network. Still, this work focuses only on the deployment of the core network service functions and does not provide any analysis study of the 5G wireless network. In Addition, the authors use two SDN layers with two different SDN controllers. In our architecture, we use a hierarchical SDN architecture with three layers (Fog, Edge and Cloud) and four SDN controllers.

B. 5G Network Slice Architectures for Automotive

Few schemes have been proposed to support 5G network slicing for the automotive vertical. Campolo *et al.* [14], [15] proposed a framework that is made of a set of network slices representing different Vehicle-to-Anything (V2X) use case categories. The framework makes use of NFV and SDN

technologies to provide different functionalities while meeting the requirements of the various applications. One of these slices is dedicated to autonomous driving. It relies on V2V communication mode as the radio access technology and uses the core network, using a centralized SDN architecture, to provide functions such as authentication, authorization and subscription management. Moreover, it deploys an application server at the network edge to help vehicles in processing 3D maps and building an augmented vision beyond their visual perception. Nevertheless, authors of [14], and [15] did not provide a mathematical framework to investigate the main requirements of autonomous driving, which are reliability and ultra-low latency. Moreover, the authors in [14] proposed to use only one slice to handle autonomous driving services, which is insufficient due to the important modules of the autonomous driving services (localization, perception, planning, and global system management). These modules need to be virtualized and scheduled using different service slices.

In our previous work [16], we have proposed a hierarchical SDN architecture for vehicular fog (HSVF). HSVF is based on a hybrid SDN control plane that reinforces centralized and decentralized management. To handle specific tasks and issues related to vehicular networks, HSVF couples SDN technologies with hierarchical and decentralized cloud architectures (i.e., fog and edge computing). To evaluate the effectiveness of HSVF, we examined a relevant case study of scheduling electric vehicles energy demands. The performance evaluations demonstrated that HSVF incurs short end-to-end delay and completion time compared to existing approaches.

From the reviewed papers, supporting the autonomous driving application in 5G given its inherent requirements (i.e., end-to-end reliability and ultra-low latency) needs further investigation in terms of core network management and orchestration, resource allocation and service differentiation.

III. Autonomous Driving Service and Function Slicing

Autonomous vehicles require four basic functions: localization, perception, planning and system management. To ensure efficient service access to these functions, we normalize them into service slices.

A. Localization Service Slice

The localization service is a crucial component of an autonomous driving system since it allows for finding optimal paths and controlling the vehicle motion. Nowadays, GPS devices are widely used as localization systems; nevertheless, the raw data position of GPS devices cannot be directly used for autonomous driving system since the quality of the GPS position can be significantly affected by satellite signal conditions. Still, GPS data can be combined with vehicle motion sensors data (wheel-speed sensors, gyros, accelerometers, and magnetic sensors) [17], environment perception data [18] and digital maps [19] to provide a more accurate positioning information. To this end, a cooperative model based information fusion service slice (CIFS) is used for the localization system.

B. Perception Service Slice

The perception service offers real time information on nearby environments using numerous types of sensor components such as cameras, radars, and laser scanners. Radars and laser scanners can detect and track static and dynamic obstacles, using moving object tracking algorithms while vision systems identify various visual objects based on vision-based object detection and classification algorithms. For an optimal perception, a machine-learning-based scheme can be employed to perceive the visual objects. The data of the detected and identified objects are forwarded to an operation center and used by several perception algorithms to handle various situation calculation of the autonomous driving system. In order to handle an ultra-low latency perception system, we implement the perception functionalities as a perception service slice (PSS).

C. Planning Service Slice

Planning algorithms for autonomous driving systems are used in order to provide safe and reliable maneuvers under various driving situations. In this paper, we focus on three principal steps for planning a trip of an autonomous vehicle. The global routing step finds the fastest and safest way to go from the initial position to the target position. A digital map management system and a data searching algorithm are essential for an ultra-low latency routing [20]. The behavior reasoning step evaluates the driving situation and controls the overall behavior of the autonomous vehicle based on the global route and perception information. The local motion can be generated in the local planning stage based on the global route and the determined behavior. The generated local motion should avoid static and dynamic obstacle collisions for safe autonomous driving. The goals of the planning service are to perform reliable autonomous driving and to achieve several missions in real time. Behavior reasoning performs a rule-based decision process based on finite-state machines (FSMs) for a global driving strategy based on the localization service and uploaded perception data. Integrated in the decision process, the rule is predefined to follow real-time traffic regulations (e.g., lane keeping, following traffic lights, and keeping under speed limits) and to accomplish various tasks.

D. Global System Management Service Slice

A global system manager is important for supervising the entire autonomous driving system. Basically, the system manager performs three principal functions: (i) Driving mode management: is divided into three modes, i.e., manual, run, and pause. In the manual mode, the vehicle is manipulated by the driver; in the run mode, the vehicle drives by it autonomous. If the system operator pushes the emergency stop switch or a system fault occurs, the driving mode is changed to the pause mode; (ii) The fault management system: this function monitors the status of all modules for safe driving. If the status of some critical modules for autonomous driving do not get updated for a certain period of time, fail management algorithms, which are implanted in each module, control and

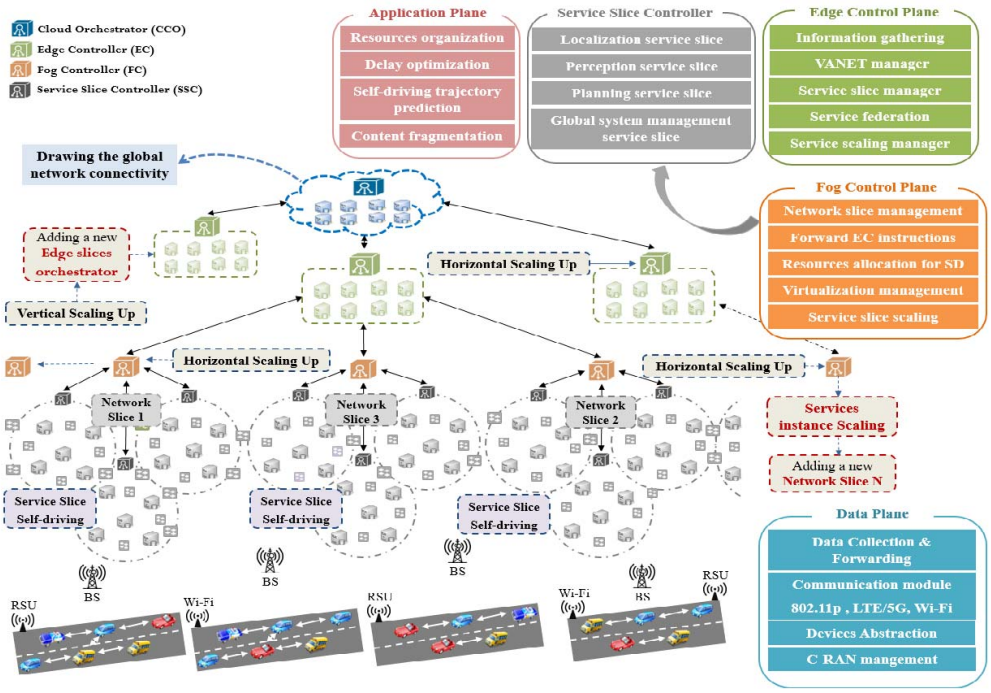


Fig. 1. 5G-slicing-enabled scalable sdn core network architecture (i.e., SliceScal).

set the state of the system to fail and change the autonomous mode to the pause mode; and (iii) The human–machine interface (HMI): contains a functional interface and a display system that shows the status of the vehicle, the localization, the trajectory, and the nearby object information. To satisfy the ultra-low latency and high reliability requirements of autonomous driving modes, the global system management service is implemented as a service slice (GMSS).

IV. Distributed Scalable SDN Core Network

A. Features of 5G-Enabled SDN Core Network Architecture

Reliable and ultra-low latency of autonomous driving service functions and requests over wireless networks alone is not sufficient to guarantee the required end-to-end reliability. The core network has also to be considered for service slice management to support these requirements. Indeed, both autonomous driving and best-effort traffics will have to travel through a complex network while competing for resources during transmission, buffering, and computing. In order meet the desired requirements of autonomous driving systems, traffic generated by autonomous vehicles has to be properly marked, identified, and prioritized at all the network elements (both wireless and core network).

In this paper, we propose a new vehicular core network based on a distributed and scalable SDN architecture called SliceScal (see Fig. 1). SliceScal uses OpenFlow, which is a communication protocol between the control plane and the network infrastructure. The SDN controllers can modify the forwarding rules for each flow that are stored in the form of flow tables. This is to maintain the priority of the critical flows with respect to other flows (e.g., best-effort traffic). In Addition, a central cloud orchestrator generates a complete isolation between different flows and service slices, and applies the necessary forwarding rules to ensure that the QoS of autonomous driving is maintained throughout the network. In a network with multiple flows requiring different levels of QoS, a distributed SDN can allow dynamical control of their Key Performance Indicators (KPIs) based on the effective demands generated by each of these flows.

B. Core Network Design

The 5G core vehicular network architecture defined in this work considers a distributed scalable SDN, it uses a stable clustering technique based on the deployment of fog cells. This technique maintains the data transmission and control instructions to be used when the central cloud orchestrator is not accessible. In addition, by distributing the control over different fog cells, we can reduce network congestion and message-control overhead.

We use 3GPP's 5G system for our vehicular network [21]; however, we introduce it as part of the global architecture for NFV virtualization. In particular, all vehicular network functions are considered to be slices that can run over standard hardware. The slices may be moved across different locations subject to the requirements of the service providers. Using this architecture, our goal is to describe the role of softwarization and service slicing from the autonomous driving perspective. This includes software-defined traffic routing decisions in the 5G radio access and defining SDN controllers' duties in the core network. In general, service slicing rules have to be enforced in the physical infrastructure to satisfy a satisfactory level of QoS for autonomous driving systems, starting from the selection of most suitable radio access nodes and up to managing traffic and service requests in the transport vehicular network.

We believe that autonomous driving the use of 5G slicing based on a distributed and scalable SDN core network is crucial for autonomous driving function services to ensure appropriate traffic isolation while guaranteeing ultra-low latency and reliability. Fig. 1 illustrates the SliceScal core network. From bottom up, our architecture includes one data access plane and three control levels: fog control plane, edge control plane, and cloud control plane. The southbound API between the control plane and data plane is implemented by extending OpenFlow [16] while the northbound API is provided to the applications as a high-level abstraction to communicate with the different controllers.

1) Data Plane (RSUs, 5G BSs, and Vehicles): The data plane includes all data transmission network devices. The main functions of the data plane are data collection, data quantization and data forwarding to the control plane through the southbound API. The data plane supports Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Infrastructure (I2I) communications using different networking technologies (e.g., DSRC, 5G, Wi-Fi, and IEEE 802.11p). The data plane is built as a hierarchical overlay fog network to eliminate the heterogeneity of existing vehicular networks. All vehicles, Road Side Units (RSUs), and 5G Base Stations (BSs) are abstracted as SDN devices and equipped with OpenFlow software switches. These software switches can be further categorized into the mobile data plane (i.e., vehicles) and stationary data plane (RSUs, 5G BSs). RSUs, BSs, and other edge equipment can communicate with micro-data centers located in the fog to deliver different services. We assume that SDN devices are capable of virtualization and can host virtual machines (VMs) that are able to install software updates on the SDN devices.

2) Control Planes (FC, EC, CC): SliceScal implements three control levels: edge control plane, fog control plane, and cloud control plane. The edge control plane maintains the status of all the OpenFlow software switches and is responsible for making packet forwarding instructions based on the OpenFlow control channel. The service slice management and policy enforcement functions for vehicular network are carried out by the Edge Controller (EC), which is also in charge of traffic routing and service slice scaling to route data to its intended destination. The Fog control plane is deployed to address the problem of frequent handovers that may occur considering the high mobility and the massive traffic between RSUs, BSs and vehicles. In this regard, we deploy a distributed control plane over different fog cells. Fog controllers (FC) are located in some of the servers of each fog cell. Each FC is connected with RSUs, BSs and vehicles via other OpenFlow switches and servers, which form a distributed fog based hierarchical architecture. To guarantee an appropriate planning of service slices and an adequate forwarding rules in the core network, the FC collaborate with the EC during the session initiation of each service slice. Indeed, the FC configures the fog cell transport network elements with the forwarding rules from the EC. Each fog cell is organized as clusters where each cluster covers a service slice for autonomous driving. Each cluster is controlled by a service slice controller (SSC) that manages the different flows arriving from the wireless access network. The service slice controller controls and manages different autonomous driving functions (presented in Section III): localization service, planning service, perception service, global system management. To ensure efficient and reliable service access to these functions, we propose to normalize them into service slices. Finally, the cloud control plane is responsible for constructing the global network connectivity by aggregating information received from all fog and edge controllers and generating control information based on rules and strategies from the application plane. The cloud control plane contains the SDN cloud orchestrator (CO), responsible for gathering information and constructing the global network connectivity. It can also be programmed to make automated decisions about the network in case of traffic congestion, faulty devices or security problems.

3) Application Plane: Based on application requirements, rules and strategies are generated by CO for ECs and by FCs for the wireless vehicular network. The application plane contains a set of applications and services like resource organization, demands management, delay optimization, and trajectory prediction. The application plane interacts with each FC to forward required information to the control plane. Additionally, in order to improve service access efficiency of autonomous vehicles, we consider the content fragmentation module as a service that is provided at the application plane. It requires the help of the autonomous-driving trajectory prediction module to enhance data dissemination while offering real time information about nearby services.

V. Network and Service Slicing System Model

Ensuring and end-to-end service is among the main goals of 5G networks. In this paper, we investigate the end-to-end performance of multi-service slicing in 5G networks based on a distributed and scalable SDN architecture. We begin by outlining the corresponding system model, then detail our function slicing model.

A. System Model

We analyze an interesting use case in vehicular networks where end-to-end communications delay is a critical metric. Indeed, we focus on autonomous driving vehicles with intensive computing tasks for real-time traffic detection and driving decision making. In this specific scenario, a softwarized 5G infrastructure enables large data transfer from various sensors on autonomous vehicles (e.g., cameras, sensors, ultrasonic radar…) to an operation center. Since the requirements for serving autonomous driving vehicles in terms of data rate and delay are hardly supported by microwave systems, the use of 5G mmWave cellular is considered here as a primary option for the autonomous vehicle. Other radio technologies are used as backup [22].

1) Environment and Mobility Model: We assume a typical urban setting similar to Central Paris. An urban outdoor environment is projected on a plane $\mathbb{R}^2$ where the roads are modeled using the Manhattan Poisson line process (MPLP) [23]. The road distribution model is featured by two unit-density homogeneous Poisson processes ψ_x, $\psi_y \subset \mathbb{R}^2$ along the

x-axis and the y-axis, respectively. At each point of these processes, an avenue (West-East direction) or a street (North-South direction) grows infinitely along the x-axis and y-axis. The total street width is w_s and there are n lanes in each direction, each having the width of w_{Lane}. We model a straight segment of a street with left-side traffic.

We assume a test scenario with a number m of autonomous vehicles denoted by AV are traveling at a random speed S_{AV} that follows the probability density function (PDF) of $f_{S_{AV}}(x)$. The height of the vehicles is denoted by h_{AV} and all antennas are placed on their roofs for better channel conditions. Also, we assume that we have a number m' of manually driven vehicles denoted by V assumed to be deployed randomly in the street. We denote by l the length of one manually driven vehicle V, its PDF is given by $f_{l_V}(x)$. The manually driven vehicles are traveling at a random speed S_V that follows the PDF of $f_{S_V}(x)$.

The vehicles are positioned on lane i following a Poisson process with intensity denoted by γ_i. d denotes the inter-vehicle distance in line i, $f_{d_i}(x)$ designates the PDF of d_i. All vehicles are assumed to have a constant width given by W and drive along the center of their lanes.

2) Heterogeneous Wireless Access Network: In SliceScal, we propose to use the concept of heterogeneous vehicular access network. This concept is proposed to satisfy the communication requirements of not only autonomous driving service slices (i.e., safety message) but also best-effort traffic (i.e., non-safety message) for both autonomous driving vehicles and manually driven vehicles. Thus, due to high mobility and the dynamic change of vehicular network topology, it is difficult to provide satisfactory services through only a single wireless access network, such as dedicated short range communication (DSRC) or third generation 4G Long Term Evolution (LTE) networks or Wi-Fi [24].

Therefore, SliceScal architecture uses a heterogeneous wireless access networks which integrate four wireless technologies: (a) dedicated short range communication (DSRC), (b) cellular mmWave network, (c) cellular microwave network (LTE), and (d) non-3GPP microwave network (Wi-Fi). DSRC is used to provide V2V and V2F communications. Cellular mmWave base stations (BSs) are assumed to be deployed alternately on the left and the right sides of the buildings on a street at a constant height of h_{BS} with a distance of d_{BS} between one another. For microwave access, we assume that both LTE and Wi-Fi connectivity is always available via road side units (RSUs). Principally, the LTE network covers the entire city and maintains a constant SNR level via adequate power control. We also assume that there is always at least one Wi-Fi access point within the coverage area around the autonomous vehicle and we consider fixed-power transmissions, where the effective spectral efficiency is determined by true distance between the vehicles and the access point [25].

B. BSs Logical Layers for Function Slicing

In this paper, the function slicing is realized by a new logical layers implemented at each BS. As we shown in Fig. 2, the BS is consisted by four principal logical layers: control layer, resource management layer, virtualization layer and interface layer.

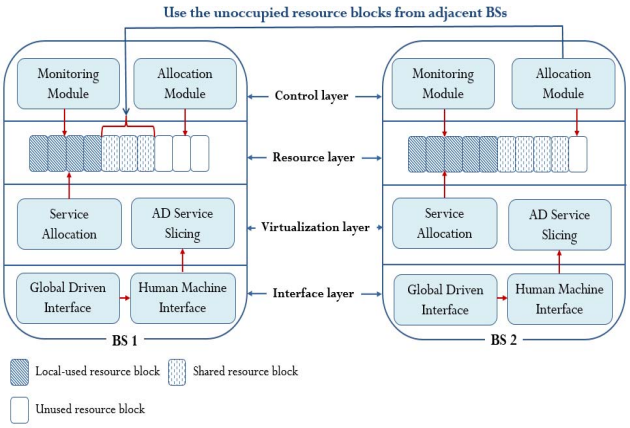


Fig. 2. Logical layers for autonomous driving function slicing.

1) Control Layer: Includes two function modules, monitoring and allocation. The monitoring module takes charge of monitoring the number of occupied resource blocks at each BS coverage area. In case the number of occupied resource blocks reaches the available number of blocks at the BSs and the new requirements of resource blocks are still needed by autonomous driving service slices, the allocation module will schedule the unoccupied resource blocks from adjacent BSs to be used by the corresponding BSs (see Fig. 2). This is to improve the efficiency of resource utilization in 5G slicing vehicular network.

2) Virtualization Layer: The main function of the virtualization layer is to organize different autonomous driving service slices and optimize the handling process for prompt response from the service slices. In addition, the virtualization layer allocates resource blocks at the location of each BS to satisfy the requirements of critical autonomous driving service slices.

3) Resource Management Layer: Manages the resource blocks at the location of BSs. Originally, BSs resource blocks are allocated to a specific autonomous driving service slice. Thus, when the BSs resource management layer cannot satisfy the location requirement of service slices, the virtualization layer can provide more resource blocks from the associated control layer by sharing unoccupied resource blocks from adjacent BSs. It is the resource management layer duty to allocate these shared blocks in order to meet the requirements of different service slices.

4) Interface Layer: Provides the interface for different types of the predefined autonomous driving functions. The interface layer covers the human–machine interface (HMI) of the global system management service slice (GMSS). The main function of BS interface layer is to separate the BS resources with different types of autonomous driving service slices. Based on the BS interface layer, the relationship of different types of autonomous driving service slices is declining by separating the resource requirements with the type of service slice.

VI. Ultra-Low Latency Mathematical Framework

In this section, we investigate the latency performance of mmWave transmission based on 5G slicing for vehicular autonomous driving service slicing.

A. Propagation Latency of Autonomous Service Slices

The propagation latency is the duration between the time the service was requested by the autonomous vehicle and the time service slices are accessed at the BS. In our work, we focus on the millimeter wave (i.e., mmWave) wireless transmission for communications between autonomous vehicles and BSs. According to our mobility model, BSs are uniformly distributed along the roads. We denote by $d_{\mathrm{AV}}^{\mathrm{BS}}$ the distance between an autonomous vehicle AV and its base station BS which has PDF of $\mathrm{f}_{\mathrm{d}_{\mathrm{AV}}^{\mathrm{BS}}}(\mathrm{x})$. First, we study the mmWave LoS blockage for mmWave BSs.

1) mmWave LoS Blockage: We consider the mmWave LoS blockage model presented in [26] and [27]. Our main interest is to determine the LoS blockage distance and the LoS time duration of an autonomous vehicle. As in [26], we consider two special scenarios as follows:

- A road with single lane: Consider the non-blocked interval caused by other vehicles in a randomly chosen lane. Let $h_{\mathrm{V}}^{\mathrm{i}}$ denotes the height of a vehicle at lane i, assuming that not all vehicles block the LoS path of a given autonomous vehicle [27]. The probabilities that a randomly chosen vehicle blocks the LoS path of the target service request of an autonomous vehicle link is given by [26] as follows

$$P_V^i = \Pr\left\{h_V > h_V^i\right\} = \int_{h_V^i}^{\infty} f_{h_V}(x)dx, \quad i = 1, 2, \ldots \tag{1}$$

where $f_{h_V}(x)$, $x > 0$ is the PDF function of the vehicle height.

We assume that the vehicle heights are randomly distributed. Thus, the number of vehicles between two consecutive vehicles that block the LoS path of an autonomous vehicle follows a geometric distribution with parameters $G_i = \frac{1}{\gamma_E^i}$, $i = 1, 2, \ldots$, and $\gamma_E^i = P_V^i \cdot \gamma_i$, where $\gamma_{\mathrm{E}}^{\mathrm{i}}$ is the effective intensity of vehicles in lane i blocking the LoS path. The PDF of the distance between two vehicles blocking the LoS in lane i denoted by $f_{d_{LoS}^i}$is given by [26] as follows

$$f_{d_{LoS}^i}(x) = \sum_{i=1}^{\infty} G_i \left[f_{l_V}(x) f_{d_i}(x)\right] \tag{2}$$

By observing the possible structure of the LoS interval in a single lane, and from Eq. (1) and Eq. (2), the contribution of LoS interval components decrease exponentially as the number of vehicles between two vehicles blocking the LoS increases. Thus, the LoS blockage distance in lane i can be approximated by the exponential distribution of the effective density of vehicles in the same lane. As in [26], the effective density of vehicles in lane i is given by

$$\gamma_i = \mathrm{G_i} \int_0^{\infty} f_{\mathcal{W}}(x)dx \int_0^{\infty} f_{S_V^i}(x)dx \tag{3}$$

- *A road with multiple lanes:* consider a road with multiple lanes with possible vehicles blocking the LoS. To calculate the blockage interval duration caused by the vehicles in multiple lanes, the authors in [27] propose to use the M/G1/∞ queuing system, where the service time distribution corresponds to time spent by a blocker in the LoS blockage lane. Assuming that the vehicles deployed across the lanes are independent. According to [27], the CDF of time in the LoS blocked interval with i lanes is the same as the distribution of a busy period in M/G1/∞ queuing system. The CDF of time in the LoS blocked interval with i lanes is given by the minimum of the LoS time intervals in i lanes as follows

$$F_{T_{LoS}}(x) = 1 - \prod_{i=1}^{n} \left[1 - F_{T_{LoS}^i}(x)\right], \tag{4}$$

Let T_V^* be the subjective blockage time caused by a single vehicle in all lanes. Based on the analysis of M/G1/∞ queuing system given in [27], the blockage time distribution can be approximated by using the mean waiting times $E\left[T_{LoS}\right]$ and $E\left[T_V^*\right]$ of the M/G1/∞ queuing system. The approximation of the mean is given in [26], [27] as

$$E\left[T_V\right] = \frac{E\left[T_{LoS}\right] E\left[T_V^*\right]}{E\left[T_{LoS}\right] - E\left[T_V^*\right]} \tag{5}$$

2) Path Loss of mmWave Wireless Link: For now, we study the path loss of wireless mmWave link between the autonomous vehicle and the BS. To this end, the orthogonal frequency division multiplexing (OFDM) scheme is assumed to be adopted by 5G mmWave BSs for accessing autonomous driving service slices from vehicles. Considering the great attenuation of mmWave wireless transmission in a long range, the fast fading and interference from adjacent BSs are ignored in our model. We denote by ϑ the threshold of signal-to-noise ratio (i.e., SNR) at receivers. The autonomous driving service slices are correctly received by receivers when the SNR of receivers is larger than ϑ. By applying the fitted model in [28], the path loss of wireless link between the autonomous vehicle and the BS is expressed as

$$PL[dB](d) = \alpha + \beta 10\log_{10}(d) + \varepsilon \tag{6}$$

where d is the distance in meters, α and β are the least square fits of floating intercept and slope over the measured distances (30 to 200 m). Thus, the values of α and β are varied according to the used frequencies and the statistical model for the large scale parameters based on the NYC data in [29]. The values of α and β can be founded in [28]. ε is the wireless link noise given by the Gaussian distribution with the zero mean and variance σ^2, (i.e., $\varepsilon \sim \left(0, \sigma^2\right)$. We assume that only one service slice is configured to transmit per time slot. Hence,

the probability that one service slice is correctly revived by a BS is given by

$$P_{acc} = P(PL \leq p_{tx}(dB) - \vartheta(dB) - \tau_0 \mathcal{W}(dB) \quad (7)$$

where $p_{tx}(dB)$ is the transmission power at BSs and vehicles, τ_0 is the noise power density, $\mathcal{W}(dB)$ is the bandwidth of wireless links. By substituting Eq. (6) into Eq. (7), P_{acc} is expressed as follows

$$P_{acc} = \frac{1}{2}\left(1 + f_{er}(\frac{\delta(d)}{\sqrt{2\sigma}})\right) \quad (8)$$

where,

$$\delta(d) = p_{tx}(dB) - \vartheta(dB) - \tau_0 \mathcal{W}(dB) - \alpha + \beta 10\log_{10}(d) \quad (9)$$

$f_{er}(x)$ is the error function.

By taking into account the blockage time of LoS calculated in Eq. (5), the propagation latency of one autonomous service slice is derived as

$$\mathbb{T}_{Prop} = \frac{t_{slot} - E[T_V]}{P_{acc}} = \frac{2(t_{slot} - E[T_V])}{1 + f_{er}\left(\frac{\delta(d)}{\sqrt{2\sigma}}\right)} \quad (10)$$

where t_{slot} is the duration of the transmission slot.

B. Handling and Tail Latencies of Autonomous Driving Slices

So far, we assume that each BS contains a number of computation resource blocks that are configured as the primary computation resources for handling autonomous driving services slices. We assume that every BS is configured with the same number of computation blocks, denoted by CB_{BS}. Rationally, the autonomous driving service slice handling speed and waiting time in each queue can be improved by increasing the number of computation blocks. As mobility of autonomous vehicles (AVs) on roads is random, the number of AVs in the coverage area of every BS is different; consequently, the arrival rate of AVs service slice at every RSU is different. In this case, the service slice handling process at the BS is assumed as a *GI/M/1* queuing system. Nevertheless, the process of service slice handling at the BS is non-Markovian.

As proved in [30], in *GI/M/1* queuing system, there exists Markov chains imbedded in the queue length process (number of requests in the queue). So, the autonomous driving service slices handling latency can be analysed using *GI/M/1* queuing system. In addition, we consider the tail latency of each service slicing by deriving the waiting time in each queue. Let $t_0 = 0$ be the arrival time of the first task, and the inter-arrival times $u_n = t_n - t_{n-1}(n \geq 1)$ have the general distribution function (GDF) denoted by $f(u)$. With the Laplace transformation, we have

$$\psi(\lambda) = \int_0^\infty e^{-\lambda u} df(u)(\lambda > 0) \quad (11)$$

where λ is the arrival rate. The service time denoted by $\{\chi_n, n \geq 1\}$ is exponentially distributed with rate $\mu(\mu > 0)$ and the distribution function of service times at every BS is given by

$$G(u) = 1 - e^{-\mu u}, \quad (u \geq 0) \quad (12)$$

Considering the *GI/M/1* queuing system, the average service time of autonomous driving service slices at BSs is determined by the number of resource blocks. To simplify our analysis, we denote by $\frac{1}{\mu_0}$ the average service time when a service slice is serviced by a resource block, and we denote by $\frac{1}{CB_i\mu_0}$ the average service time when a service slice is serviced by a BS with CB_i resource blocks. Based on the *GI/M/1* queuing system and the proposed mathematical framework, the handling latency of autonomous driving service slices at the BS can be expressed using by the following theorem.

Theorem 1: The cumulative distribution function (CDF) of one of the autonomous driving service slices interval time arriving at BS_i $(i = 1, 2, \ldots, m)$ is

$$F(t) = 1 - \frac{e^{-\gamma_i L\left(1 - e^{-\lambda_s t}\right)} - e^{-\gamma_i L}}{1 - e^{-\gamma_i L}} \quad (13)$$

where γ_i is the density of vehicles in lane i calculated in Eq. (3) and $\lambda_s t$ is the transmission rate of autonomous driving service slices generated by one autonomous vehcile.

Proof: For the time duration of handling one service slice $t \geq 0$, we have

$$P\{T_0 > t + s | T_0 > s\} = P\{T_0 > t\} = e^{-\lambda_s t} \quad (14)$$

where T_0 is the service slice interval time generated by an autonomous vehicle.

We assume that we have a number n of AVs in the coverage area of a BS. The probability that the service slice time arriving at the BS, denoted by T^s_{BS}, should less than or equal to the time duration t and is given by

$$\begin{aligned} P\{T^s_{BS} \leq t\} &= 1 - P\{T^s_{BS} > t\} = 1 - PC_1 = 1 - \frac{P\{C_2\}}{\sum_n P\{C_3\}} \\ &= 1 - \frac{\sum_{n=1}^{\infty}\left(e^{-\lambda_s t}\right)^n \frac{(\gamma_i L)^n}{n!} e^{-\gamma_i L}}{1 - e^{-\gamma_i L}} \end{aligned} \quad (15)$$

where C_1, C_2, and C_3 are defined as follows

C_1 : there is no service slice generated in the coverage of the BS in the time slot $(0, t)$.

C_2 : n AVs do not transmit service slice to the associated BS.

C_3 : there are n AVs on the road that transmit service slice to the associated BS.

From Eq. (13) and Eq. (14), the CDF of one autonomous driving service slice interval time arriving at the BS can be derived by Eq. (15), and then Theorem 1 is proved.

As a result, we can derive the PDF of autonomous driving service slice time arriving at the BS by

$$f_A(t) = \frac{\lambda_s \gamma_i L e^{-\gamma_i L + \gamma_i L e^{-\gamma_i L} - \lambda_s t}}{1 - e^{-\gamma_i L}} \quad (16)$$

Our main interest is to determine the handling latency (i.e., response time) and the tail latency (i.e., waiting time) for SliceScal core network architecture. Our first goal is to derive the state transition probabilities for the embedded chain (in our

case study, a state represents the total number of service slice requests in the queuing system including service slice requests, if any, in the BS computation block). A transition from state i can increase to at most $i+1$ and can decrease by up to i during an arrival epoch. As shown earlier, the inter-arrival times $u_n = t_n - t_{n-1}$ $(n \geq 1)$ have the general distribution function (GDF) denoted by $f(u)$, and the distribution function of service times at every BS is given by Eq. (15). Since service times are exponentially distributed with rate $\mu(\mu > 0)$, the number of autonomous driving service requests that leave the queue in t seconds is distributed as a Poisson random variable with parameter μt. Hence, we have the following two cases:

- *The queue system does not go empty during an arrival epoch:* the probabilities of state transition can be written as

$$P_{i,i+1-j} = \int_0^\infty \frac{e^{-\mu t}(\mu t)^j}{j!} f(t)dt, \quad j = 0, 1, \ldots, i, \quad (17)$$

- *The queue system goes empty during an arrival epoch:* the probabilities of state transition can be expressed as

$$\begin{aligned} P_{i,0} &= \int_0^\infty \left(1 - \sum_{k=0}^{i} \frac{e^{-\mu t}(\mu t)^k}{k!}\right) f(t)dt, \\ &= \int_0^\infty \left(\sum_{k=i+1}^{\infty} \frac{e^{-\mu t}(\mu t)^k}{k!}\right) f(t)dt \quad i = 1, 2, \ldots \quad (18) \end{aligned}$$

Eq. (17) and Eq. (18) indicate that the embedded stationary probabilities π_i^e, $i = 0, 1, \ldots$, satisfy global balance. This implies that

$$\pi_i^e = \sum_{j=0}^{\infty} \pi_j^e P_{j,i} = \sum_{j=i-1}^{\infty} \pi_i^e \int_0^\infty \frac{e^{-\mu t}(\mu t)^{j+1-i}}{(j+1-i)!} f(t)dt \quad (19)$$

To solve Eq. (19), we must satisfy the normalization condition for the stationary embedded probabilities. To do so, we use the method guessing technique. A simple guess is that the distribution is geometric. Hence, we have

$$\pi_i^e = (1-\beta)\beta^i, \quad i = 0, 1, \ldots \quad (20)$$

Substituting Eq. (20) to Eq. (19), we have

$$\begin{aligned} (1-\beta)\beta^i &= (1-\beta) \sum_{j=i-1}^{\infty} \beta^j \int_0^\infty \frac{e^{-\mu t}(\mu t)^{j+1-i}}{(j+1-i)!} f(t)dt \\ &= (1-\beta) \int_0^\infty e^{-\mu t} \beta^{i-1} \sum_{j=i-1}^{\infty} \frac{(\mu t \beta)^{j+1-i}}{(j+1-i)!} f(t)dt \\ &= (1-\beta) \int_0^\infty e^{-\mu t(1-\beta)} \beta^{i-1} f(t)dt \quad (21) \end{aligned}$$

Eq. (21) can be written in terms of Laplace tranforms by canceling common terms

$$\beta = \mathcal{L}(\mu(1-\beta)) \quad (22)$$

Thus, we have two cases:

- If the solution of Eq. (22) is unique and yields $(0 < \beta < 1)$, then the stationary distribution of the embedded stationary state probabilities for the proposed *GI/M/1* queuing system at each BS during the time of arrival are given by

$$\pi_i^e = (1-\beta)\beta^i, \quad i = 0, 1, \ldots \quad (23)$$

- If the traffic intensity $\rho = \lambda/\mu$ is less than 1, then it can be shown that β, as determined by Eq. (22), is uniquely determined and is less than 1.

Now, we can easily determine the tail latency and the handling latency time for the used queue at each BS. First, to derive the density of the response time, we use the fact that the geometric sum of exponential random variables is exponentially distributed, as illustrated in Theorem 1. Thus, if an arriving service slice request x finds i requests in the queue, then x's response time equals the sum of $i+1$ independent exponential random variables with parameter μ. Consequently, x's response time is Erlang with parameters $(i+1, \mu)$. This implies that the response time for handling one autonomous driving service slice request at a BS is given by the following probability density function (PDF)

$$\begin{aligned} f_H(t) &= \sum_{i=0}^{\infty} (1-\beta)\beta^i \frac{\mu(\mu t)^i e^{-\mu t}}{i!} \quad (24) \\ &= \mu(1-\beta) e^{-\mu t(1-\beta)} \quad (25) \end{aligned}$$

This means that it is exponentially distributed with parameter $\mu(1-\beta)$, implying that the mean handling latency (i.e., response time), including can be given by

$$\mathbb{T}_{Hand} = E[H] = \frac{1}{\mu(1-\beta)} \quad (26)$$

And the tail latency (i.e.,waiting time) is given by

$$\mathbb{T}_{Tail} = E[W] = \frac{\beta}{\mu(1-\beta)} \quad (27)$$

Eq. (25), Eq. (26), and Eq. (27) show that the main difficulty in determining the stationary statistics for our proposed queuing system is to determine β by solving Eq. (25). As proved in [31], this is usually solved using numerical techniques.

Example: Consider an arrival rate of two autonomous driving service slice at a BS with parameter $(2, \lambda)$. To determine β, we must solve Eq. (25) which in this example is expressed as $\beta = \left(\frac{\lambda}{\mu(1-\beta)+\lambda}\right)^2$. This yields a cubic equation $\beta^3 - 2(1+\alpha)\beta^2 + (1+\alpha)^2\beta - \alpha^2 = 0$ where $\alpha = \lambda/\mu$. We observe that to obtain the handling and the tail latencies of an $G_k/M/1$ queue system, we need to solve for the roots of a polynomial of degree $k+1$. This can be done numerically.

Finally, from Eq. (10), Eq. (26) and Eq. (27), the total latency of one AV service slice request at each BS is given by:

$$\begin{aligned} \mathbb{T}_{Total} &= \mathbb{T}_{Prop} + \mathbb{T}_{Tail} + \mathbb{T}_{Hand} \\ &= \frac{2(t_{slot} - E[T_{v_O}])}{1 + f_{er}\left(\frac{\delta(d)}{\sqrt{2\sigma}}\right)} + \frac{\beta}{\mu(1-\beta)} + \frac{1}{\mu(1-\beta)} \quad (28) \end{aligned}$$

VII. Autonomous Driving Resource Slicing Management Algorithm

In this paper, we assume that there exists K_{BS} BSs in a coverage of a Fog cell. When we adopt the autonomous driving service slicing, we use a matrix $\mathbb{M}_{K_{BS}*K_{BS}}$ to describe the multiplexing of resource blocks at each Fog cell. The elements of $\mathbb{M}_{K_{BS}*K_{BS}}$ are denoted by $\varphi_{i,j}, (1 \leq i \leq K_{BS}, 1 \leq j \leq K_{BS})$. When $i \neq j$, $\varphi_{i,j}$ represents the ratio of resource blocks at the $j-th$ BS which is scheduled for the $i-th$ BS. When $i = j$, $\varphi_{i,j}$ is the ratio of resource blocks at the $j-th$ BS used for the local service slices. In order to ensure the local service slice handling at the BS, we propose to configure an upper bound $\varphi_{i,j,Max}$ to restrict the ratio of local BS resource blocks used for adjacent BSs (i.e., $0 \leq \varphi_{i,j} \leq \varphi_{i,j,Max} < 1$ when $i = j$). autonomous driving. Since BSs can use resource blocks of adjacent BSs to handle service slices of local Fog cells, handling the latency of autonomous driving service slice can be enhanced and improved to ensure autonomous vehicles requirements. Yet, when requests are to be forwarded to adjacent BS, incured propagation latency needs to be considered in our model. To simplify derivation of the propagation latency of service slice forwarding, we assume that the propagation latency between BS_i and the adjacent BS_j, denoted by $\mathbb{T}_{prop_i^j}$, can be calculated by Eq. (11). Thus, the total latency of service slice forwarding from BS_i to BS_j is given by

$$\mathbb{T}_{Total_{FW}} = \mathbb{T}_{Prop} + \mathbb{T}_{prop_i^j} + \mathbb{T}_{Tail} + \mathbb{T}_{Hand} \tag{29}$$

Based on SliceScal architecture, we propose a autonomous driving resource slicing management algorithm. The proposed algorithm schedules the resource blocks of each Fog cell as illustrated in Algorithm 1. When an autonomous driving service slice request arrives at BS_i, the associated Fog cell controller checks whether the number of local resource blocks CB_i can satisfy the latency requirement $\mathbb{T}_{req}$ of service slice; if yes (i.e., $\mathbb{T}_{Total_{FW}} < \mathbb{T}_{req}$), the Fog cell controller decreases the number of local resource blocks of BS_i until $\mathbb{T}_{Total_{FW}} = \mathbb{T}_{req}$. In this case, we denote by CB_i' the number of resource blocks which satisfies $\mathbb{T}_{Total_{FW}} = \mathbb{T}_{req}$. The remaining number of resource blocks $CB_i'' = CB_i - CB_i'$ at BS_i is put into the virtualization network layer which can be scheduled by the resource management layer using the Fog cell controller. If no, (i.e., $\mathbb{T}_{Total_{FW}} > \mathbb{T}_{req}$), the Fog cell controller increases the number of resource blocks scheduled by BS_i, until $\mathbb{T}_{Total_{FW}} = \mathbb{T}_{req}$. The additional number of resource blocks denoted by $CB_i^{req} = CB_i' - CB_i$ is added from the virtual network layer and scheduled by the resource management layer to support the autonomous driving service slice handling at the BS_i.

Algorithm 1 Autonomous Driving Resource Slicing Management Algorithm

Input : Base station BS_i, number of resource blocks $CB_i, \mathbb{M}_{K_{BS}*K_{BS}}, \mathbb{T}_{req}$
Output: Update resource slicing blocks

```
Initialize M_{K_BS*K_BS}, CB_i, T_req, φ_{i,j,Max};
Generate autonomous driving requests with T_req;
for i = 1 to K_BS do
    Calculate T_{Total_FW} with CB_i by Eq.(33);
    while (T_{Total_FW} < T_req) do
        Update CB_i'' by decreasing CB_i;
        CB_i'' = CB_i − CB_i';
        Put CB_i'' into to the virtual layer;
        Calculate T_Hand and T_Tail with CB_i';
    end
    while (T_{Total_FW} > T_req) do
        Update CB_i^req by increasing CB_i;
        CB_i^req = CB_i' − CB_i
        Add CB_i^req to CB_i from the virtual layer;
        Calculate T_Hand and T_Tail with CB_i';
    end
end
```

VIII. Performance Evaluation

In this section, we present a simulation-based evaluation of the proposed system. We examine the effectiveness of SliceScal and the 5G slicing model for autonomous driving. We develop an evaluation platform using network simulator NS-3, vehicles traffic simulator Veins-SUMO [32], and OpenFlow SDN controller. We employ a realistic urban scenario that may serve as a representative setup for a variety of possible deployment configurations. The road topology was obtained from OpenStreetMap, filtered, formatted, and converted into a SUMO network file.

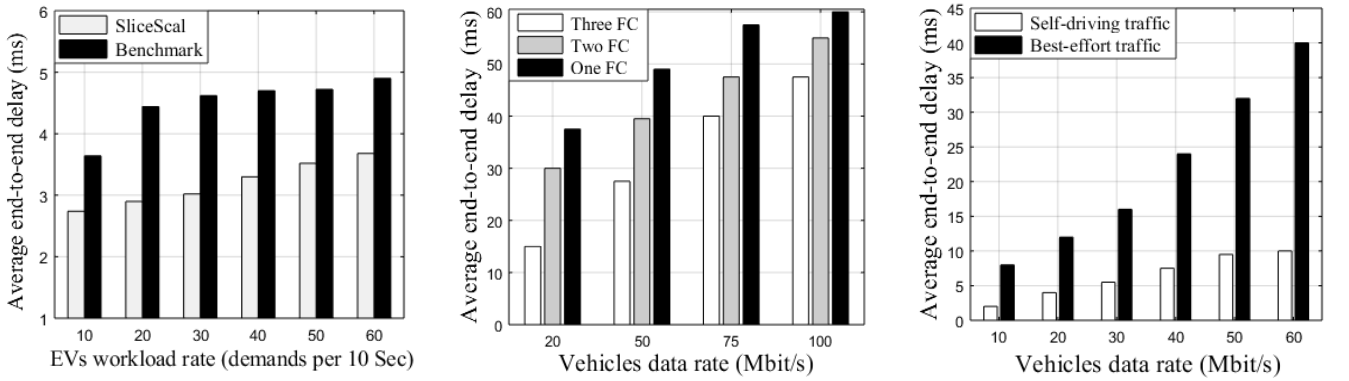
A. Parametrization of Experimentation Scenario

We model a 4-lane bidirectional street of 20 m width in the city of Paris. Each lane has a width of 3.65 m. We assume the Tesla Model S form factor with the dimensions of 4.979 m, 1.964 m, and 1.445 m as the autonomous vehicle. The number of AVs follow a uniform distribution from 2 to 10 vehicles. Following the data from [33], we allow the length of other vehicles, to follow a Gamma distribution with a mean of 4.5 m and a variance of 0.5 m^2. As for the height, we assume that it adheres to a Gamma distribution with a mean of 1.7 m and a variance of 0.3 m^2. The speed of vehicles (including autonomous vehicles) is uniformly distributed between 20 km/h and 80 km/h, which is representative of dense urban setting.

The mmWave radio propagation follows the default 3GPP protocol [34], where the case of blockage is modeled as NLoS conditions. According to 3GPP release 14, in case of blockage, the mmWave radio access is capable of establishing an alternative NLoS path that is currently non-blocked. We also adopt the typical parameters of mmWave cellular. Specifically, we set the transmission power to 25 dBm and the center frequency to 73 GHz with 2 GHz of bandwidth. Antennas' gain for BS and autonomous vehicles is assumed to be 15 dB and 10 dB, respectively. The noise floor is set to -80 dB.

B. SliceScal Implementation

To demonstrate the ultra-low latency approach for autonomous driving advocated in this paper, we construct a



(a) SliceScal Vs Benchmark (b) Scaling of fog controllers (c) Autonomous driving Vs Best-effort

Fig. 3. Performances of slicescal core network.

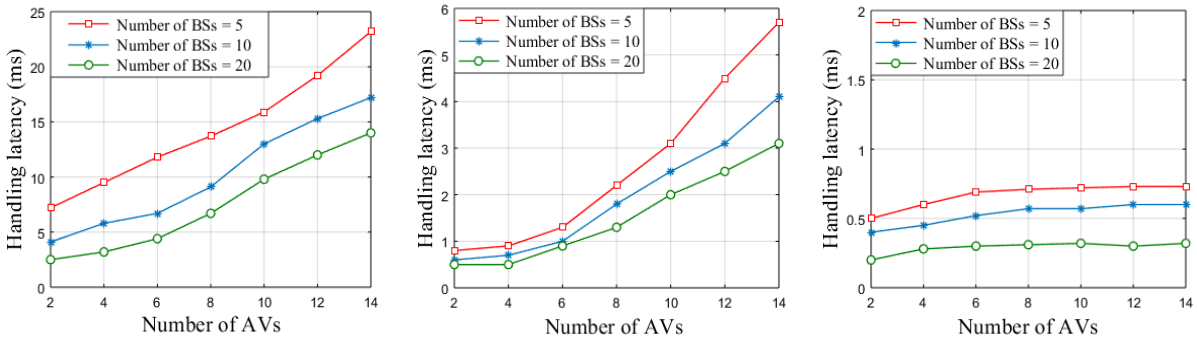
simple but representative proof-of-concept implementation of SliceScal core network. In our test implementation, we use two Linux servers implemented on two DELL OptiPlex 7050 with 32GO of RAM and Intel core i7-7500T CPU 3.75GHz. Each Linux machine acts as the SDN edge controller (EC) In each Linux machine, we implement five VMs, three of them act as SDN fog controllers (FC). We use Mininet 2.1.0 with OpenFlow 1.3 for the SDN FC data plane as it is more scalable and provides a realistic environment. The two remaining VMs are used as hosts to either generate or receive data traffic. The generated UDP-flows are created with iperf [35] and represent either autonomous driving traffic or best effort traffic (other vehicles traffic.

Further, we employ the OpenDaylight (ODL) SDN controller to manage FC switches. The communication between the ODL controller and the switches is based on the OpenFlow protocol. In order to have a distributed control plane, the topology of our core network has been implemented using POX [36] as in [1]. To maintain the priority of the critical autonomous driving flows with respect to other flows (e.g., best-effort traffic), FC maps a flow onto a priority queue. Relying on each FC controller, we assign the data flows initiated by the VMs hosts to the queues [38], [39]. We distinguish the critical autonomous driving traffic and the best-effort traffic using the destination IP address in the packets. The minimum (guaranteed) and maximum service rates need to be maintained in each of the queues to avoid disturbance from the best-effort traffic toward the critical autonomous driving traffic since the sum of both data flows represent the full capacity of the link.

C. Performance Results

To evaluate the average end-to-end delay, we make vehicles generate traffic at different data rates. Fig. 3(a) illustrates the average end-to-end delay of over SlcieScal against a benchmark scheme that does not use the SDN approach. As vehicle's data traffic increases, we observe that SlcieScal outperforms the benchmark topology by (30% to 45%) in term of average end-to-end delay. We also evaluate the average end-to-end delay with respect to generated traffic flows when scaling the SliceScal by using different number of distributed fog controllers (FCs) (i.e., One FC, two FC, three FC). Fig. 3(b) shows that the topology with three FCs incurs the shortest average end-to-end delay compared to the other topologies. Indeed, when using only one FC, the controller gets overwhelmed as the generated traffic per vehicle increases, leading to packet drops and high communication overhead. However, by partitioning the vehicular network into sub-networks (i.e., fog cells and clusters), where each fog cell has an SDN controller, we endorse the load balancing capability. In fact, fog controllers are responsible for only managing the vehicles located in their corresponding geographical area. In Fig. 3(c), we compare the average end-to-end delay of autonomous driving traffic and best-effort traffic in the core network. We can observe that autonomous driving traffic incurs is shorter end-to-end delay compared to best-effort traffic. This is because FC maintains the priority of the critical autonomous driving flows with respect to other flows (e.g., best-effort traffic) by mapping them into a priority queue. Results presented in Fig. 4 demonstrate that the SlcieScal core network is suitable for scheduling and transmitting critical traffic (i.e., autonomous driving traffic) especially when using 5G wireless network and service slicing.

Fig. 4 shows the handling latency of autonomous vehicles service requests in three different scenarios. In the first scenario (see Fig. 4(a)), we run simulations without service slicing in BSs. In the second scenario (see Fig. 4(b)), we run simulations with service slicing while in the third scenario (see Fig. 4(c)), we run simulations with service slicing along with the slice management algorithm (SliceMan). We observe that the handling latency increases with the increase of AVs' density. The case of 20 BSs incurred the shortest handling latency in the three scenarios compared to the 5 and 10 BS cases. We also observe in that using service slicing reduces the handling latency by 60% (see Fig. 4(b)) compared to scenario 1 (see Fig. 4(a)). Finally, the use of SliceMan algorithm helps reduce the handling latency by 90% (see Fig. 4(c)) compared to scenario 1 (see Fig. 4(a)). This is due to the effectiveness of the proposed service slice management through allocating the required resource blocks to service slices using the virtualization network layer with the help of SSC.



(a) Without service slicing (b) With service slicing (c) With SliceMan algorithm

Fig. 4. Comparison of handling latency of autonomous vehicles service requests.

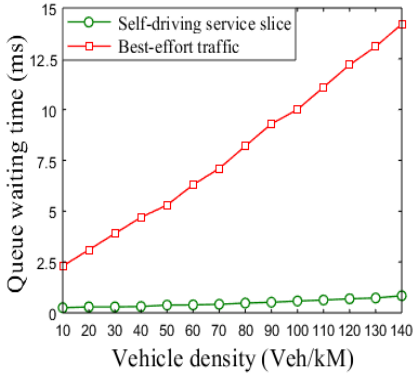


Fig. 5. Tail latency comparison.

Fig. 6. Latency of service slice.

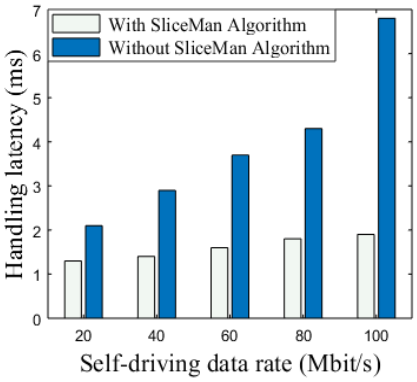
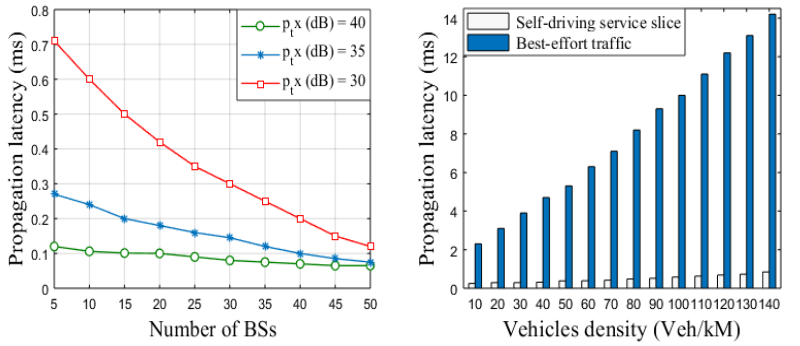


Fig. 7. Autonomous driving over slicman.

To evaluate the efficiency of our proposed solution and the scalability of SliceScal over realistic scenarios, we run simulation with different number of vehicle densities (from 10 Veh/kM to 140 Veh/kM). Fig. 5 shows the effectiveness of the proposed *GI/M/1*queuing system implemented inside each BS resource blocks. We observe that the tail latency (i.e., waiting time) of autonomous driving service slice is shorter than the best-effort traffic by 80%. We also observe that the waiting time increases (i.e., from 2.5 ms to 14.5 ms) for best-effort traffic with the increase in vehicle density while it remains almost stable for critical autonomous driving service slice (i.e., from 0.3 ms to 1 ms). Fig. 6 shows the handling latency comparison between autonomous driving service slice, autonomous driving without service slice, and best-effort traffic as a function of vehicles' density. We observe that the handling latency increases with the increase in vehicles' density for best-effort traffic (i.e., from 25 ms to 57 ms) and autonomous driving without service slice (i.e., from 15 ms to 20 ms) while it stays almost autonomous driving stable for autonomous driving service slice (i.e., between 0.5 ms and 1 ms).

To evaluate the SliceMan algorithm, we generate different traffic flows with varying autonomous driving data rates. We run two scenarios; one that uses the SliceMan algorithm on scheduling autonomous driving service slice and the other which does not use the SliceMan algorithm. We observe from Fig. 7 that the SliceMan algorithm reduces the handling latency by 75% when scheduling autonomous driving traffic flows of 100 Mbit/s autonomous driving. This is because the SliceMan algorithm offloads the number of resource blocks between the BSs using the virtualization network layer of service slices which can be scheduled by the appropriate resource management layer.

Fig. 8 shows the assessment of in the propagation latency over the BSs. Fig. 8(a) depicts the propagation latency with respect to the number of implemented BSs in the road considering different values of vehicles' transmission power (i.e., $p_{tx}(dB)$). As we know in VANETs, a high density of BSs reduces the propagation latency. Thus, we can see in Fig. 8(a) that when $p_{tx}(dB)$ is fixed, the propagation latency decrease even if RSUs' density increases of. Also, increasing the transmission power, decreases significantly the propagation latency. Fig. 8(b) depicts the propagation latency with respect to the number of vehicles deployed in the road autonomous driving. We observe that when the vehicle density increases, the propagation latency of best-effort traffic increases significantly (i.e., from 2.1 ms to 14 ms) while



(a) Transmission power (b) Autonomous driving service slicing

Fig. 8. Propagation latency comparison.

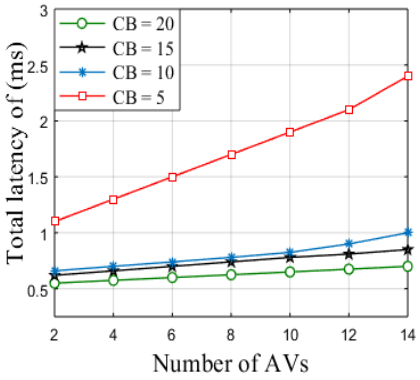


Fig. 9. SliceMan algorithm evaluation.

it remains stable for autonomous driving service slice (i.e., from 0.3 ms to 1 ms and is 85 % less than the best effort's) autonomous driving. This is because the autonomous driving service slice uses the 5G mmWave access point to transmit AVs critical demands while the best-effort traffic is offloaded over the other access points (Wi-Fi and LTE).

Finally, Fig. 9 illustrates the total latency of autonomous driving service slice (i.e., handling latency and propagation latency) with respect to AVs density considering different numbers of assigned computation blocks per BS. We observe that the total latency decreases with the increase in the number of computation blocks. The reason is threefold. 1) the use of *GI/M/1* queuing system for analyzing the handling latency of AVs demands over autonomous driving service slices; 2) the use of the SliceMan algorithm that, reduces the handling latency significantly by scheduling the resource blocks at each BS (see Fig. 8); and 3) the use of SliceScal as a core network enhances the reliability of autonomous driving service slice and copes with scalability and mobility issues of vehicular networks.

IX. Conclusion

The automotive industry has recently shifted from developing advanced vehicles to safe and comfortable ones, which stimulates the development of new intelligent vehicles with autonomous driving control. However, reliable and efficient communication is extremely important to guarantee the safety and comfortability of autonomous deriving. Thus, the challenges are provisioning end-to-end reliability and ultra-low latency for high-rate critical autonomous driving traffic between a server in the core network and the highly-mobile autonomous vehicles. A solution to this challenge is very complex and requires efficient orchestration of network functionality at different levels (i.e., wireless access network, core network). The orchestration should deal with the spontaneous changes of the serving access point according to the user mobility pattern as well as highly dynamic channel conditions.

In this work, we first introduced a distributed and scalable SDN core network architecture called SliceScal designed specifically for our use case (i.e., autonomous driving service slicing). We then presented the essential autonomous driving functions that need to be sliced. Next, we presented our ultra-low latency mathematical framework where we define a propagation latency model and a handling latency based on GI/M/1 queuing system. Afterwards, we present an autonomous driving resource slicing management algorithm for scheduling computation resource blocks for handling different slices. The simulation results have demonstrated that the SlcieScal core network is suitable for scheduling and transmitting critical autonomous driving traffic especially when using 5G wireless network and service slicing. In addition, the use of SliceMan algorithm reduce the handling latency by 90% through allocating the required resource blocks to service slices using the virtualization network layer. Finally, the proposed service slicing methodology can be further employed to analyze particular (beyond-)5G scenarios for autonomous driving, that involve different deployments, various user mobility models and desired programmable connectivity policies to optimize network deployment configurations.

References

[1] W Mohr, "5G empowering vertical industries," 5G-PPP, ERTICO, EFFRA, EUTC, NEM, CONTINUA Networld2020 ETP, Eur. Commission Eur. ICT Ind., Belgium, Brussels, Belgium, White Paper, 2016. [Online]. Available: https://ec.europa.eu/digital-single-market/en/blog/5g-empowering-vertical-industries-0

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[3] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[4] J. Erfanian, "5G white paper," Next Gener. Mobile Netw. Alliance, Frankfurt am Main, Germany, White Paper 1.0 NGMN Board, Feb. 2015, vol. 1.0.

[5] *Applying SDN Architecture to 5G Slicing*, document ONF TR-526, Jan. 2016.

[6] N. Lang, M. Rüßmann, J. Chua, and X. Doubara, "Making autonomous vehicles a reality: Lessons from Boston and beyond," Automot. Mobility, Boston Consulting Group, Munich, Germany, Tech. Rep., Oct. 2017.

[7] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[8] R. Miller, "Autonomous cars could drive a deluge of data center demand," Data Center Frontier, Atlanta, GA, USA, May 2017.

[9] J. Prados, A. Laghrissi, M. Bagaa, T. Taleb, and J. M. Lopez-Soler, "A complete LTE mathematical framework for the network slice planning of the EPC," *IEEE Trans. Mobile Comput.*, to be published.

[10] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5G network slices to support industrial Internet and to shape next-generation smart factories," *IEEE Netw.*, to be published.

[11] S. Costanzo, I. Fajjari, N. Aitsaadi, and R. Langar, "A network slicing prototype for a flexible cloud radio access network," in *Proc. 15th IEEE Annu. Consum. Comput. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2018, pp. 1–4.

[12] L. Velasco *et al.*, "An architecture to support autonomic slice networking," *J. Lightw. Technol.*, vol. 36, no. 1, pp. 135–141, Jan. 1, 2018.

[13] L. Ma, X. Wen, L. Wang, Z. Lu, and R. Knopp, "An SDN/NFV based framework for management and deployment of service based 5G core network," *China Commun.*, vol. 15, no. 10, pp. 86–98, Oct. 2018.

[14] C. Campolo, A. Molinaro, A. Iera, and F. Menichella, "5G network slicing for vehicle-to-everything services," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 38–45, Dec. 2017.

[15] C. Campolo, A. Molinaro, A. Iera, R. R. Fontes, and C. E. Rothenberg, "Towards 5G network slicing for the V2X ecosystem," in *Proc. 4th IEEE Conf. Netw. Softw. Workshop*, Montreal, QC, Canada, Jun. 2018, pp. 400–405.

[16] D. A. Chekired, M. A. Togou, and L. Khoukhi, "Hierarchical wireless vehicular fog architecture: A case study of scheduling electric vehicle energy demands," *IEEE Veh. Technol. Mag.*, vol. 13, no. 4, pp. 116–126, Sep. 2018.

[17] K. Jo, K. Chu, and M. Sunwoo, "Interacting multiple model filter-based sensor fusion of GPS with in-vehicle sensors for real-time vehicle positioning," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 329–343, Dec. 2012.

[18] I. P. Alonso, Y. Mulgaonkar, N. Michael, and V. Kumar, "Accurate global localization using visual odometry and digital maps on urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1535–1545, May 2012.

[19] K. Jo, K. Chu, and M. Sunwoo, "GPS-bias correction for precise localization of autonomous vehicles," in *Proc. IEEE IV*, Jun. 2013, pp. 636–641.

[20] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Jan. 2011.

[21] *System Architecture for the 5G System*, document TR 23.501, v1.4.0, 3GPP, 2017.

[22] "5G and e-Health," 5G-PPP, Eur. Commission Eur. ICT Ind., Brussels, Belgium, White Paper, Feb. 2015.

[23] F. Baccelli, M. Klein, M. Lebourges, and S. Zuyev, "Stochastic geometry and architecture of communication networks," *Telecommun. Syst.*, vol. 7, nos. 1–3, pp. 209–227, Jun. 1997.

[24] K. Zheng, Q. Zheng, H. Yang, L. Zhao, L. Hou, and P. Chatzimisios, "Reliable and efficient autonomous driving: The need for heterogeneous vehicular networks," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 72–79, Dec. 2015.

[25] O. Galinina, A. Pyattaev, S. Andreev, M. Dohler, and Y. Koucheryavy, "5G multi-RAT LTE-WiFi ultra-dense small cells: Performance dynamics, architecture, and trends," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 6, pp. 1224–1240, Jun. 2015.

[26] V. Petrov *et al.*, "Achieving end-to-end reliability of mission-critical traffic in softwarized 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 485–501, Mar. 2018.

[27] M. Gapeyenko *et al.*, "On the temporal effects of mobile blockers in urban Millimeter-wave cellular scenarios," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10124–10138, Nov. 2017.

[28] G. Zhang, T. Q. S. Quek, M. Kountoursi, A. Huang, and H. Shan, "Fundamentals of heterogeneous backhaul design—Analysis and optimization," *IEEE J. Sel. Areas Commun.*, vol. 64, no. 2, pp. 876–889, Feb. 2016.

[29] T. S. Rappaport *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[30] A. Cobham, "Priority assignment in waiting line problems," *J. Oper. Res. Soc. Amer.*, vol. 2, no. 1, pp. 70–76, 1954.

[31] N. U. Prabhu, *Foundations of Queueing Theory*. Norwell, MA, USA: Kluwer, 1997.

[32] (Jan. 2019). *Veins Vehicle in Network Simulations: The Open Source Vehicular Network Simulation Framework*. [Online]. Available: http://veins.car2x.org/

[33] *Car Dimensions of any Make and Model in the European Market*. Accessed: Jan. 2019. [Online]. Available: http://www. automobiledimension.com/

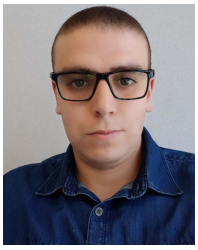
[34] *Channel Model for Frequency Spectrum Above 6 GHz (Release 14)*, document 3GPP TR 38.900 V14.3.1, 3GPP, 2017.

[35] iPerf. *The Ultimate Speed Test Tool for TCP, UDP and SCTP*. Accessed: Jan. 2019. [Online]. Available: https://iperf.fr/

[36] SDN Hub. *POX Controller Tutorials*. Accessed: Jan. 18, 2018. [Online]. Available: http://sdnhub.org/tutorials/pox/

[37] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Decentralized cloud-SDN architecture in smart grid: A dynamic pricing model," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1220–1231, Mar. 2018.

[38] D. A. Chekired and L. Khoukhi, "Smart grid solution for charging and discharging services based on cloud computing scheduling," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3312–3321, Dec. 2017.



Djabir Abdeldjalil Chekired (S'17) received the B.S. degree in computer science and the M.S. degree in computer networks and distributed systems from the University of Science and Technology Houari Boumediene (USTHB), Algiers, Algeria, in 2010 and 2012, respectively. He is currently pursuing the the joint Ph.D. degree in computer science with the Environment and Autonomous Networks Laboratory (ERA), University of Technology of Troyes, France, and the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. His research interests include new cloud computing design for smart grid systems and analysis in green electric vehicles networks, smart grid energy management, fog-SDN architectures, and the Internet of Things.

Mohammed Amine Togou received the B.S. and M.S. degrees in computer science and computer networks from Al Akhawayn University, Ifrane, Morocco, and the Ph.D. degree in computer science from the University of Montreal, Canada. He is currently a Postdoctoral Researcher with the Performance Engineering Laboratory, Dublin City University, Dublin, Ireland. His current research interests include connected vehicles, SDN–NFV, technology enhanced learning, IoT, smart cities, and machine learning. He is currently involved in the Horizon 2020 EU Project NEWTON.

Lyes Khoukhi (M'09) received the Ph.D. degree in electrical and computer engineering from the University of Sherbrooke, Canada, in 2006. In 2008, he was a Researcher with the Department of Computer Science and Operations Research, University of Montreal, Canada. He is currently a Professor in computer science with the University of Technology of Troyes, France. He has authored or coauthored over 100 publications in reputable journals, conferences, and workshops. His research interests include IoT, cloud, smart grids, and mobile communication protocols. He has participated as the general chair or a TPC member of many conferences.

Adlen Ksentini received the Ph.D. degree in computer science from the Cergy-Pontoise University in 2005. He is an IEEE Communications Society Distinguished Lecturer. From 2006 to 2016, he was an Assistant Professor with the University of Rennes 1. In 2016, he joined the Communication Systems Department of EURECOM as an Assistant Professor. He has been involved in network slicing in the context of the EU-funded H2020 Projects 5G Pagoda and 5G-TRANSFORMER.