An Application-Driven Framework for Intelligent Transportation Systems Using 5G Network Slicing

Tiago do Vale Saraiva, Carlos Alberto Vieira Campos, *Member, IEEE*, Ramon dos Reis Fontes, Christian Esteve Rothenberg, Sameh Sorour, *Senior Member, IEEE*, and Shahrokh Valaee, *Fellow, IEEE*

***Abstract*— Vehicular networks are critical pieces in support of advanced intelligent transportation systems (ITS). These networks are formed by vehicles that can be connected to one another as well as to the infrastructure, and are subject to constant topology changes, disconnections, and data congestion. Each ITS application could have a different set of communication requirements, such as delay, bandwidth, and packet delivery ratio. Meeting these heterogeneous requirements in the complex dynamic environment of vehicular networks is a challenge. This paper develops a new framework for application-driven vehicular networks using 5G network slicing. We present the architecture of the proposed solution and design algorithms for heterogeneous traffic in a dynamic vehicular environment. Our simulations on realistic vehicular scenarios show significant improvements in network performance compared to the state-of-the-art approaches.**

***Index Terms*— Vehicular networks, network slicing, software-defined networking, application-driven networks, intelligent transportation systems.**

I. Introduction

INTELLIGENT Transportation Systems (ITS) enclose a wide range of technologies concerning modern vehicular services around safety, information and entertainment. These services include, but are not limited to, advertisements, tourist information, traffic information, and parking, and are expected to attract a considerable market in vehicular networks [1].

The related works often use the terms "vehicular applications" and "ITS applications" as synonyms, and we do the same in this article. Research on Vehicular Adhoc Networks (VANETs) has developed advanced wireless communication schemes to enable vehicular applications. For instance, Dedicated Short-range Communications (DSRC) was defined in the IEEE 802.11p standard, to improve road safety and transport efficiency. In the emerging paradigm of Vehicle-to-Everything (V2X) networks, vehicles can communicate with countless devices, including roadside infrastructure, altogether shaping a new scenario for safer, cheaper, intelligent, connected, and autonomous transportation systems [3], [20].

On the other hand, the highly dynamic environment of vehicular networks causes dramatic changes in the spatial and temporal behavior of the network topology, resulting in communication quality degradation. At the same time, several vehicular applications need their communication requirements to be addressed dynamically in such complex environments. Traditional mobile communication networks employ a "one-size-fits-all" approach to provide services to mobile devices, regardless of the communication requirements of vertical services [5]. Thus, the evolution of vehicular applications, and increasing demands and challenges imposed by vehicular environments, raises reliability concerns on VANETs, since this fixed resource allocation mode of operation is inadequate to satisfy the envisioned driving environments [4].

At the crossroads, technological advances of the fifth generation (5G) cellular networks and their related enabling technologies, include not only improved radio access networks but also Software Defined Networks (SDN), Network Function Virtualization (NFV), and Multi-access Edge Computing (MEC), among others. To meet the specific service requirements of different applications, 5G leverages the concept of *network slicing*, where a single network and compute infrastructure is used to deploy customized service slices that meet specific requirements. With network slicing, it is possible to tailor the slices for diverse and complex 5G communication scenarios [5]. Due to its potential in addressing Internet of Vehicles (IoV) requirements on ultra-low delay and high reliability among other specific applications [4], the topic of network slicing in 5G has become a growing research trend [6], [7].

Although several works address the problem of how to meet the requirements of different applications in vehicular networks, there is a demand of detailed solutions with proper evaluation work, which takes into account the multifaceted aspects of vehicular networks and applications. This article proposes a framework for the deployment of vehicular networks that dynamically meet the requirements of different ITS applications.

Manuscript received March 30, 2020; revised October 9, 2020 and April 3, 2021; accepted May 28, 2021. The Associate Editor for this article was F. Granelli. *(Corresponding author: Tiago Do Vale Saraiva.)*

Tiago do Vale Saraiva and Carlos Alberto Vieira Campos are with the Department of Applied Informatics, Federal University of State of Rio de Janeiro (UNIRIO), Rio de Janeiro 22290-240, Brazil (e-mail: tiago.saraiva@uniriotec.br; beto@uniriotec.br).

Ramon dos Reis Fontes is with the Digital Metropolis Institute, Federal University of Rio Grande do Norte (UFRN), Natal 59078-970, Brazil (e-mail: ramonreisfontes@gmail.com).

Christian Esteve Rothenberg is with the Department of Computer Engineering and Automation, University of Campinas (UNICAMP), Campinas 13083-852, Brazil (e-mail: chesteve@dca.fee.unicamp.br).

Sameh Sorour is with the School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: sameh.sorour@queensu.ca).

Shahrokh Valaee is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: valaee@ece.utoronto.ca).

Digital Object Identifier 10.1109/TITS.2021.3086064

Following the principles of 5G network slicing, we have developed methods for integrating SDN controllers so that they use the communication requirements of applications and information related to vehicles' mobility in each Radio Access Network (RAN) to meet the needs of ITS applications dynamically.

The proposal is validated using a realistic emulation approach with an urban congestion scenario, with vehicular applications having different data rate requirements. The local and global controllers acting to prioritize the application's communication flows properly. The results are compared with an approach that makes use of quality-of-service (QoS) mechanisms and another approach from the literature [20], that use SDN to manage the flows but does not prioritize application traffics. The main contributions of the proposed Framework are:

1) An algorithm for SDN controllers based on information related to different ITS applications requirements, such as priority, bandwidth, latency, and packet loss, apply global policies to prioritize the available infrastructure communication resources to meet applications KPIs. We also provide an algorithm to integrate this component with the orchestration features of a Management and Orchestration (MANO) entity that the existing solutions can implement.
2) An algorithm that uses the information about topology changes in the vehicular context obtained through the local controllers in each RAN to manage the communication flows locally to contribute to the overall solution in meeting the application requirements.
3) A proof-of-concept prototype with open source SDN controllers, switches, and OpenFlow features in an emulated environment, which is used to evaluate our Framework, performing comparisons against an approach that uses only QoS and a state-of-the-art approach presented in [20].

The rest of this paper is organized as follows. Section II provides a summary about the related work. The proposed framework is described in details in Section III. Section IV details the settings of the evaluation experiments through a proof-of-concept prototype. Section V presents the simulation results. Finally, Section VI concludes the paper.

II. Related Work

The idea of directing network behavior to meet application demands, with efficient resource utilization, is not new and has already been considered in the literature related to QoS provisioning [9]. However, the existing solutions are limited for adoption in modern environments, mainly in scenarios with vehicular networks for ITS. The proposal in [19] makes use of Application-Driven Networks (ADN), which are networks capable of providing data paths to meet communication requirements of applications dynamically. Although the work provides an architecture with control algorithms and a proof-of-concept related to vehicle driver training, the solution does not consider the specific components of vehicular networks, such as mobility and RSUs.

Some works, such as [15], [16], and [17] propose solutions to address the QoS of applications by considering the communication requirements of different applications in the dynamic environment of vehicular networks, and making use of SDN technology. On the other hand, these works do not use 5G network slicing and do not specify the necessary algorithms for the components of the proposed solutions to work correctly. These works also lack a realistic proof-of-concept experiment, considering both vehicular and network core components.

In [20], the authors analyzed the potential of Software-Defined Vehicle Networks, emphasizing the need for rethinking the traditional SDN approach from a theoretical and practical perspective, to manage communication and networking resources in the vehicular environment. Similarly, [21] considers both 5G network slicing and the dynamics of the vehicular environment, making use of a realistic proof-of-concept experiment that uses SDN in the core of the network and considers the aspects of the vehicular environment. A framework that also uses SDN to adjust network conditions to meet ITS application requirements dynamically is proposed in [18]. The problem with these works is that they do not specify the algorithms of the components of the proposed solution and do not propose a clear way to meet the communication requirements of different ITS applications dynamically.

In [13], the authors propose a solution with fog computing and simulate a proof-of-concept experiment to produce results based on traffic conditions generated by SUMO [26] and OpenStreetMap [27] to support some vehicular applications. However, the proposed solution does not deal with meeting specific requirements of vehicular applications. The authors in [22] also make use of fog computing, but also consider the need of applications. Similarly, [14] proposes a platform to provide vehicular cloud services. Nonetheless these works do not provide the algorithms for the solution of a realistic proof-of-concept experiment.

5G network slicing in a mobile SDN core network has been used in [23] to meet the requirements of the use case of autonomous vehicles application. The authors consider the dynamic of the vehicular environment and present a management algorithm for slicing autonomous driving resources. The proof of concept experiment uses realistic SDN and mobility components. As their focus is on autonomous vehicles, the proposal does not consider meeting the communication requirements of different applications dynamically.

Since our proposal deals with SDN to dynamically configure the network to meet different ITS applications' requirements, there is room to make the infrastructure even more efficient by using the Vehicular Service Cloud (VSC) technology. The proposal in [24] uses a solution that forms nearby low-latency VSC dynamically as per the needs of vehicular users.

In the same way that the 'non-radio' aspect of 5G communication (e.g., SDN and NFV) has become more important, in 6G, the learning and intelligence aspects will become crucial. They must be integrated with the communication networks [25]. This way, the emerging solutions have Artificial Intelligence (AI) approaches to deal with the ITS environment's inherent complexity. Since our proposal uses SDN and defines the integration with the different components

using the architecture and algorithms proposed, future research of approaches can provide the integration with AI solutions running in the SDN controllers using the proposed Framework. A novel general AI solution that can be adapted to autonomously dealing with the selection of the most appropriate algorithm and the necessary parameter fine-tunings is proposed in [25].

Based on the above studies, we propose that a complete solution that satisfies the requirements of ITS applications must take into account the following items:

- **Scenarios with different applications and requirements:** this is important because there is a slew of applications designed with different QoS needs for several use cases, and a preferred solution must be prepared to manage the network resources considering this reality. Examples of these use cases are cooperative collision avoidance system, augmented reality, intelligent navigation system based on real-time road conditions, autonomous driving, platooning, and 4K live video, among others [3], [16].
- **Dynamics of the vehicular environment:** since the content delivery in vehicular networks is challenging due to node movements [28], the solution must consider the mobility of the vehicles. This mobility could result in different network topologies and resource consumption affecting the applications. Thus, it is necessary to configure the network dynamically to keep meeting application requirements properly.
- **Algorithms:** The design of dynamic resource allocation algorithms is crucial for increasing the efficiency of future sliced networks, and intelligent algorithms able to forecast mobile service demands and anticipate resource reconfiguration are required [29]. Thus, the solutions must provide the necessary algorithms in order to provide the functionalities required by the applications.
- **5G network slicing and SDN:** The emerging 5G technology is foreseen as the promising solution for improving network performance and management while ensuring a high data rate and enhanced QoS [23]. Hence, the novel solutions must consider 5G network slicing and its key technologies, such as SDN.
- **Realistic proof-of-concept experiment:** Critical issues emerge when trying to put software-defined vehicular networks into practice. These include full-featured OpenFlow protocol stacks and a realistic experimental evaluation considering real/deployable code, network conditions, mobility, and overall reproducibility [20]. This way, a realistic proof-of-concept experiment considering the network core elements and the dynamics of the vehicular environment must be conducted to prove that the proposed solutions are suitable and behave appropriately in practice.

From the review of the related literature, it is apparent that several open problems require further exploration. In the present work, we aim to fill the exposed gaps by providing a detailed solution to meet different ITS applications' communication requirements dynamically. To the best of our knowledge, this paper is the first to consider both the requirements of applications and the dynamics of the vehicular networks, with proper validation, using realistic SDVN emulation, providing an experimental comparison with an approach that uses QoS and another from the literature which uses SDN without QoS. A comparison of our proposal with related work is presented in Table I.

TABLE I
OUR PROPOSAL COMPARED TO THE LITERATURE

	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	Our proposal
Realistic prof of concept experiment						✓		✓	✓		✓	✓		✓
Different applications and requirements.		✓	✓	✓	✓		✓			✓		✓	✓	✓
Dynamic of the vehicular environment	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
5G network slicing								✓	✓	✓	✓			✓
Algorithms							✓				✓			✓
Software-defined Networks			✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓

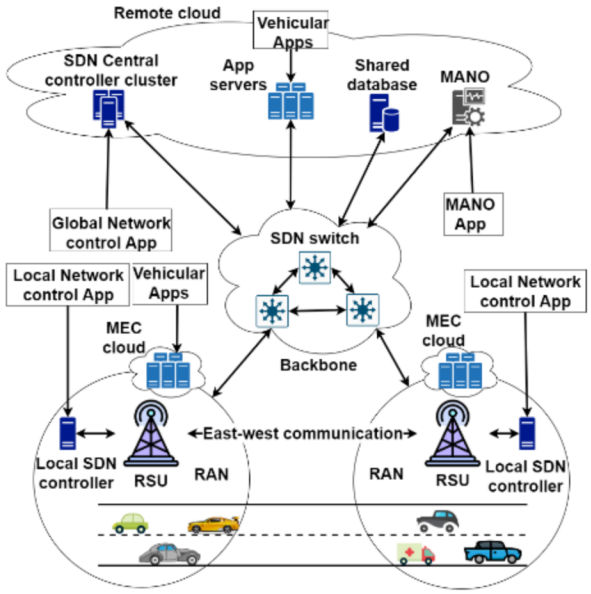


Fig. 1. Architecture of the proposed framework.

III. A FRAMEWORK FOR APPLICATION-DRIVEN VEHICULAR NETWORKS

Vehicular communication networks typically consist of vehicles with OBUs and RSUs. RSUs provide communication between the vehicles and the infrastructure of the city through cellular base stations, and we use in this paper the terms RSU and base station with the same meaning. Figure 1 depicts the components of the considered architecture in this paper, and how they are interconnected. The elements of this architecture work together with some key technologies used in 5G, namely network slicing.

Network slicing and SDN are technologies in 5G that bring significant improvements to both the radio access networks (RAN) and the core networks of mobile communications. With the concept of network slicing, 5G aims to meet diversified service requirements under the background of existing technologies [5]. Whereas the technology needed

to support the different types of slices is well understood, the implications of network slicing in terms of efficiency of network resource utilization are still not well understood [29]. Although it is difficult to determine when procedures to meet specific vehicle communications requirements will be standardized in 5G, the lack of alternatives has motivated the interest in using 5G for vehicular communications [2].

As defined by the Open Networking Foundation [30], a SDN architecture is composed of the application plane, the control plane, and the data plane. The data plane devices, which handle packet switching and traffic flow, are separated from the control plane devices, which define how the data plane should operate. The application plane is populated by application instances that represent client entities that can request some service from a SDN controller server [16], making the network management more flexible since it is possible to dynamically reconfigure the network to meet the requirements of the applications. Following a SDN-like organization, our proposed framework's architecture is detailed in sub-sections III-A, III-B, and III-C.

A. Application Plane

In a SDN there exist two types of applications, namely the end applications that are supported by the infrastructure, and the control applications that define the network behavior. In this way, these two types of applications, which will be referred to as the "vehicular" and "control" applications, respectively, will constitute the application layer in the proposed solution.

In some related works, vehicular applications interact directly with network controllers to have their communication demands met. This is typically not the best approach in a large and complex network supported by several operators, and maybe having different controllers with their APIs. In our approach, the controller devices and vehicular applications share the relevant information through a shared database, using a standard Data Manipulation Language (DML) such as Structured Query Language (SQL). So, the control level becomes transparent to vehicular applications, which can thus be indifferent to control functions and focused on providing their services.

Given the above model, each vehicular application in a smart city will have its separate algorithms to accomplish its role, and another set of common algorithms to update relevant information to all applications, such as its service subscribers and application requirements, in the shared database. In our framework, the service subscribers are the vehicles that employ the framework's vehicular applications.

As illustrated in Algorithm 1, the generic application algorithm receives new vehicle subscriptions, vehicle unsubscriptions, or updates of key performance indicators (KPIs). Subsequently, the relevant state is updated in the shared database and the Management and Orchestration (MANO) is notified about the application identifier (app_id) and modify flag (MF) changes.

Regarding the control applications, we consider three types: Global Network Control App, Local Network Control App, and MANO App. Being the applications related to the control plane, their algorithms will be detailed in Section III-C.

Algorithm 1 Generic Application Algorithm

Input: $vehicle\_subscriptions$, $vehicle\_unsubscriptions$, KPI
Output: Database Update, MANO notifications
1: Begin
2: Update in the shared database new $vehicle\_subscriptions$, $vehicle\_unsubscriptions$, and KPI changes
3: Set $MF \leftarrow 1$
4: Notify MANO about the changes, sending modify flag (MF) and the application ID (app_id)
5: End

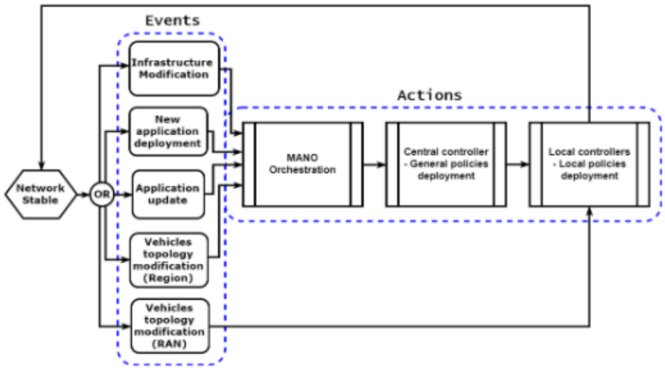


Fig. 2. Flow chart of the framework in function of changes in topology, applications or infrastructure resources.

B. Data Plane

In our framework, the data plane consists of the SDN switches that exist both in the network core and in the vehicles. They receive switching rules from the controllers of the control plane to both forward packets and prioritize vehicular applications data accordingly.

C. Control Plane

Since the objective of the proposed framework is to meet applications' requirements in a dynamic environment, it is necessary to define when the network leaves a stable state; that is when the infrastructure is not attending to applications' requirements (through KPIs). It is also crucial to determine the components to implement regain stability. As illustrated in Figure 2, the network can leave the stable state due to one or more of five reasons, all resulting in a transitory group of actions executed by the control components to bring the network to steady state again.

When there is the deployment of new base stations in the city, the implementation of a new vehicular application, or applications update, it is necessary that MANO verifies the current available infrastructure to analyze if a resource re-optimization is necessary. The above also applies when there are modifications of the topology at the level of regions, such as vehicles moving to another city. A generally more frequent scenario is that of vehicles moving between the coverage ranges of different base stations, modifying the network topology at the level of RANs, and in some cases resulting in high utilization of resources. As mentioned earlier, the network control algorithms must take necessary actions in

Algorithm 2 Management and Orchestration Algorithm

Input: app_id, AIR, ICI, VSI, KPI, MF
Output: Central controller notifications, Infrastructure orchestration, Database queries and updates
1: Begin
2: **if** $MF = 0$ **then**
3: Query the shared database and Update DS variable
4: Compute the resources for the new application, using AIR, DS, ICI, VSI, KPI, and store the result in RTI
5: Deploy the infrastructure defined in RTI
6: Update in the shared database the information of the infrastructure associated with the new application
7: Notify the Central Controller, sending MF and the identifier of the new application (app_id)
8: Update the DS variable with information updated from the shared database
9: **else**
10: Query the shared database and Update NDS variable
11: Compute changes, based on NDS and DS, and store the result in DC
12: Compute the resources for the new scenario, based on DC, and stores the result in RTI
13: Deploy the infrastructure defined in RTI
14: Update in the shared database the information of the infrastructure associated with the application
15: Update the DS variable with information updated from the shared database
16: Set $MF = 1$
17: Notify the Central Controller sending MF and app_id
18: End

all these scenarios for the network to go back to stability. The next sub-sections will illustrate our proposed algorithms to do so.

1) MANO (Management and Orchestration): MANO provides management and orchestration of physical and virtual resources in both MEC and remote clouds. As a result, dynamically it instantiates or migrates infrastructure resources, including application servers and virtual network functions, for each vehicular application, as can be seen in Algorithm 2.

The MANO algorithm uses the MF control variable to differentiate if the parameters received as input are related to a new application ($MF = 0$) or not ($MF \neq 0$). When deploying a new application, the stakeholders of the city that offers services need to submit to MANO some fundamental information, besides $MF = 0$, which are:

- Application Infrastructure Resources (AIR), which contains information about demands of network resources (e.g., network interfaces, DNS registers, NTP definition), and servers (e.g., container, HTTP, FTP).
- Infrastructure Configuration Information (ICI), which contains information such as container image ULR and IP address allocation.
- Vehicles Subscription Info (VSI), which is composed of the identification codes for each vehicle, the applications associated with them, and the geographical regions to which they belong.
- Application Identifier Code (app_id), which is unique for each application.
- Key Performance Indicators (KPI) for the applications' requirements, such as E&E Latency, Reliability, and Data Rate.

MANO also needs to verify the current status of the elements in the infrastructure under its domain. It thus queries the shared database, store the result in the DS (Database Status) variable and then uses it in conjunction with AIR, ICI, VSI, and KPI to compute the necessary resources to meet the applications' KPIs. MEC solutions (e.g., instantiating a Docker container) could be used to meet the requirements of delay sensitive applications, if the KPI of latency is lower than a specific threshold.

After the resources are computed, MANO proceeds to deploying the necessary actions at the infrastructure. Afterwards, it updates the information related to the new application, KPI, vehicles, and infrastructure in the shared database. Next, MANO notifies the central controller by sending the value of MF and the identifier of the deployed App (app_id), so that the controller proceeds as described in Subsection III-C.2 to implement the necessary global network policies. Once the deployment is done, MANO queries about the more updated data in all tables in the shared database and stores them locally in DS variable for use in future cases where $MF \neq 0$.

If MANO receives $MF \neq 0$, this is the case of an update that can be triggered by an application that (1) updates its information in the shared database; or (2) defects some modifications at infrastructure level. In these cases, MANO defines a variable named NDS (New Database Status) to store new data values, and other variable DC to save the changes. Next, MANO computes the changes and stores them in DC. With previous and difference data, MANO computes resources to the new scenario and deploys them. Once the resources are deployed/optimized, MANO updates this information at the shared database, updates the DS variable with the most updated data, and notifies the central controller with $MF = 1$ and the app_id of the application related to the updates.

In summary, the process of management and orchestration conducted by MANO consist of virtual resource instantiation and VNF placement considering the application's requirements (KPIs). This management and orchestration process can be performed in several ways, as defined by the MANO Framework [10]. In [11] is presented some technologies currently used. A robust solution that can be used to orchestrate heterogeneous virtual infrastructures (with networking and computing components) is the OpenBaton [12], which provides an engine that can be used to ensure QoS for network slices and also manage mobile edge computing using container servers deployment tools. Our framework integrates this process of resource allocation executed by the MANO with the central and local network controllers, responsible for defining the actions to be performed by the SDN components when dealing with the ITS applications' communication flows. The gain of this integration is that the controllers could configure the network components considering the infrastructure deployed by the MANO, considering the applications requirements and vehicles' mobility.

2) Central Network Controller: The central network controller is the entity that determines the general network policies for the data plane and its algorithm (Algorithm 3) has as

Algorithm 3 General Policies Algorithm

Input: MF, app_id
Output: Database Queries, General network policies deployment, Local controllers notification
1: Begin
2: **if** $MF = 0$ **then**
3: Query the shared database and update the DS variable
4: Create ARL, AVL, BSI, IAL lists, based on the DS variable
5: **for each** application **in** ARL **do**
6: Create a slice for the application and store in $SDL(app\_id)$, based in app_id, AVL, IAL, and BSI
7: Compute the global network policies to meet application requirements and store in GP, based on $SDL(app\_id)$ and ARL
8: Configure the global policies in the elements of the application slice, based on GP and $SDL(app\_id)$
9: Notify the local controllers in each RAN, sending GP
10: **else**
11: Query the shared database and update NDS variable
12: Compute the changes and store in DC, based on NDS and DS
13: Update ARL, AVL, BSI, and IAL, based on DC
14: **for each** application **in** ARL **do**
15: Update $SDL(app\_id)$ based on ARL, AVL, BSI, and IAL
16: Compute the global policies to meet application requirements and store in GP, based on $SDL(app\_id)$ and ARL
17: Configure the global policies in the elements of the application slice, based on GP and $SDL(app\_id)$
18: Notify the local controllers in each RAN, sending GP
19: End

input MF, and app_id. The outputs of the algorithm are the Database Queries, local controllers notifications, and General network policies deployment, which result in rules for configurations such as communication flow redirection and QoS. The last output is the notification to local controllers in each RAN, sending the global policies configured in the network.

When $MF = 0$, i.e., MANO finished the deployment of a new application, the central controller needs to optimize networks resources by applying the necessary policies meeting applications requirements. So, the central controller query the shared database, store the result in DS, and use it to create the variable lists of (1) applications' KPIs (ARL), (2) vehicles associated to each application (AVL), (3) IP or URL addresses and the region associated to local controllers in each RAN (BSI), (4) the infrastructure deployed by MANO per application (IAL).

For each application in ARL, the central controller defines a new network slice considering all AVL, BSI, and IAL data. In this way, each slice will include the elements, such as vehicles, servers and RSUs, related to each application. The same device could be associated with more than one slice (e.g., a vehicle subscribing to more than one application). After slice creation, the central controller computes the general policies for each slice, so as to meet the applications' KPIs, and applies them to the network (e.g., using the Openflow protocol). Examples of network policies are redirection rules and QoS queues defined by the application algorithms in the controllers. The resulting actions can be implemented with the OpenFlow protocol to set the flow entries between vehicles and application servers to meet the ITS application requirements. In the experiment, we used the information from the application servers (IP address and network connections), the information of which vehicle was associated with which application, and the application requirements (BW and priority) to define and configure, using the Ryu SDN controller with OpenFlow, the QoS queues to meet application requirements. This definition of which QoS queues and rules to configure and on which SDN switch to apply them is an example of how the solution computes the necessary resources. Afterwards, local controllers could be notified about the policies created.

If the central controller receives $MF \neq 0$, it needs to query the database to identify the changes using the $CC(NDS, DS)$ function. With these changes, it update the ARL, AVL, BSI, and IAL lists and the slices, compute the new necessary policies, and notify the local controllers.

3) Local Network Controller: The local controllers work to dynamically meet the application requirements in each RAN, following the central controller's global policies and considering the requirements of the applications in the vehicles that belong to its own RAN. The algorithm that runs in each local controller (Algorithm 4) has as input the global policies implemented by the central controller (GP), the information about KPIs that the neighboring controllers can meet ($NAKPI$), the maximum total KPI that the current RSU of the controller can meet by default (TKC), and the verification time interval (IV) for local controllers to identify the vehicles in its RANs. The value of IV can be determined using heuristics that consider several variables of the city's transportation system (e.g., holidays, accidents), even using AI approaches. TKC variable can be defined when the RSU is deployed and updated later if necessary.

Each Local controller uses $TKNA$ ("Total KPI Necessary") variable to save the resources requirements in its RAN in a given time-frame, while AV ("Availability") is used to store the available resources in the RAN.

First, the controller clears $TKNA$ ("Total KPI Necessary") variable, identifies vehicles in its RAN, saves the result in RV ("RAN Vehicles"). For each vehicle in RAN, the controller query the shared database and calculates the total sum of KPIs of the applications that the vehicle is a subscriber and store the result in $D(Vi)$. The sum of requirements for all vehicles in RAN is saved in $TKNA$. The controller also queries the shared database to update AV, which is the balance of resources redirected from or to the local RSU by its neighbors. To get its total balance of resources, the RSU controller calculates $AKPI = TKC - TKNA + AV$.

When $AKPI > 0$, the local controller send $AKPI$ to inform its neighbors, via an east-west communication API, that it has idleness of resources that could be used (e.g., available bandwidth in the links that connect it with the network backbone), and waits IV seconds to recheck the RAN status. Each message with $AKPI$ that the controller receives stores the reported information locally in a file ($NAKPI.file$).

If $AKPI < 0$, it means that the local RSU cannot accomplish the KPIs and, this way, it uses the information of the priority of the applications in the shared database to create prioritization indexes (API and $MPAI$) to manipulate the flows properly. Next, it uses the $NKPI.file$ to verify if the neighboring RSUs can handle part of the demand. If the

Algorithm 4 Local Controllers Algorithm

Input: GP, $NAKPI$, TKC, IV
Output: Local network control policies configurations, Available balance notifications to neighbors, Database queries and updates
Begin
$TKNA = 0$
Identify all vehicles in RAN and save its information in RV
Query the shared database and update AV
for each $NAKPI$ message received from neighbors **do**
    Store in the local file $NAKPI.file$ the resource balances received from the RSU neighbor
for each vehicle Vi in RV **do**
    Query from the shared database the total sum of KPIs from the applications which Vi is subscriber and store the result in $D(Vi)$ and add to $TKNA$
Calculate the local RSU balance ($TKC-TKNA+AV$) and store the result in $AKPI$
if $AKPI < 0$ **then**
    Query priority information of applications from the shared database and store in AI
    Define API Max level of priority class and store in $MAPI$, based on AI
    Set $API = 1$ to initiate with less priority applications
    while $API < MAPI$ **do**
        Set $BS\_Id = NULL$
        Verify in $NAKPI.file$ if there is some neighbor that can meet the KPI of the class API Application and store the result in BS_Id
        if then$BS\_id \neq null$
            Compute the policies to redirect flows of class API application to neighbor with the ID equal to BS_Id and store in LPR
            Configure LPR in the network
            Update the shared database, based on LPR
            Set $API = API + 1$
        else
            Compute the policies to limit the traffic of class API application according to GP and save in LQP
            Configure LQP in the network
            Update the shared database, based on LQP
else
    If there are left over resources, store the value in $AKPI$ and send it to neighbors
Wait IV time to a new verification
Back to Begin

result is positive, the neighboring unit's identifier that can attend is saved in BS_id. So, the local controller configures the policies to redirect the applications' traffic (LPR) and updates the information of redirection in the shared database. If no one neighboring unit can attend the additional demand, the local controller deploys the policies to limit the traffic of the application (LQP) based on the QoS determined by the Global Policies. In this case, the database is updated with the traffic-limit information since it will be necessary to correctly calculate the balance of resources.

IV. Performance Evaluation

We now turn our attention to the practical realization[1] and performance evaluation of the proposed framework based on the Mininet-WiFi wireless network emulator [31], a fork of the well-known Mininet emulator [32]. Our implementation considers four classes of application priority: classes A, B, C, and G. Class A is reserved to applications that have ultra-low latency requirements and must use a MEC solution, whereas classes B and C are more delay tolerant. In the case of limited resources in the communication with remote servers, Class B application has more priority than C. Lastly, class G represents common applications which have no priority nor specific requirements.

The role of central controller in the network is implemented using the Ryu SDN controller,[2] which is a top-rated software that supports the latest versions of the OpenFlow protocol and has a very active community [33]. The SDN controller uses the information of vehicles, RSUs, and applications requirements stored in the shared database, which was developed using MySQL, to configure global QoS policies in the SDN switches of the network.

Periodically, the local controller identifies the vehicles in the RAN of each RSU, queries the shared database to determine the applications associated with these vehicles, and queries the shared database to obtain the information about the applications' requirements. After that, these requirements are summed to obtain the accumulated requirements for each vehicle in the RAN. By adding the calculated demands for each vehicle in the RAN, the local controller can compute the total requirements that the RSU needs at a given time.

The implementation considers the need to prioritize the application data rate KPI appropriately. Thus, the local controller queries the shared database to check for information about traffic redirected by the RSU or to it. If the RSU has redirected or limited some data traffic in the past, the related data rate's value needs to be added to the final balance. On the other hand, if it receives some traffic redirected, the value must be subtracted from its balance.

When the RSU is congested, the local controller triggers some actions to prioritize the applications of Class B as opposed to those of Class C having lower priority. That said, local controllers verify whether their local list of vehicles with Class C applications is not empty and some neighboring unit has a balance to deal with the related data rate appropriately. If both verifications return true, these vehicles' flows are redirected (one at a time, as long as necessary) to the neighboring RSU unit. The information about the redirection is updated by the local controller in the shared database. These registers are eliminated when the vehicles are no longer connected to the RAN, avoiding distortions in balance results.

If the RSU is congested and cannot redirect Class C flows, the controller tries to redirect Class B application flows. If this is not possible, it limits Class C's application flows locally by associating them to a default queue of non-priority traffic. If this non-priority queue already contains traffic from other non-priority applications (i.e., Class G), the local controller also blocks this traffic to not compete with the limited Class C priority traffic.

In the evaluation environment, when a vehicle leaves the coverage area of an RSU, the association timeout is about

[1]Source code, data, configuration information and supporting documentation enabling experiment reproducibility are available at the public project repository: https://github.com/saraivacode/framework_its_sdn

[2]https://osrg.github.io/ryu/

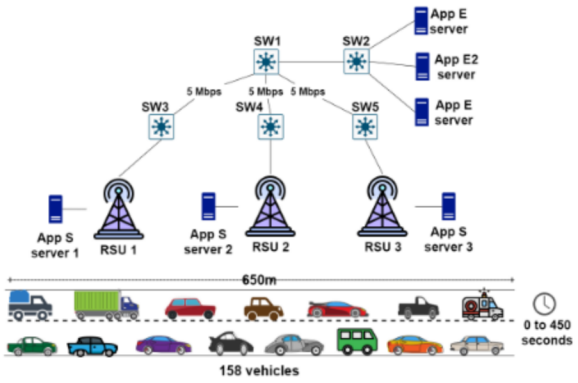


Fig. 3. Evaluation scenario representing a traffic jam in a dense urban road.

20 seconds. Thus, the local controller has to wait at least 20 seconds to confirm a possible traffic congestion before act to prioritize the applications. As will be shown in the analysis of the results, this have some impact on the results.

TABLE II

SUMMARY OF THE PARAMETERS USED IN THE PERFORMANCE EVALUATION SCENARIO

Parameter	Value
Number of vehicles	158
Number of RSUs	3
RSUs range	250m
Backbone SDN switches	5
Application servers	6
Propagation Model	Log Distance
RAN MAC layer	IEEE 802.11g
Data rate per vehicle associated with priority applications	2 Mbps
Number of applications	4
RSUs Upload link BW	5 Mbps
Emulation time	450 seconds

TABLE III

APPLICATIONS CHARACTERISTICS

Applications	Use	Data rate KPI	Protocol	Port	Priority class
S	Safety	500 Kbps	UDP	5002	A
E	Efficiency	500 Kbps	UDP	5003	B
E2	Entertainment	1 Mbps	UDP	5004	C
G	Generic	500 Kbps	UDP	5005	G

A. Evaluation Scenario

Representing an ordinary situation that occurs in the traffic of most cities around the world, the evaluation scenario consists of a traffic jam, where there is a time-increasing density (i.e., vehicles per m^2). As illustrated in Figure 3, during the whole 450 seconds of evaluation time, 158 vehicles move in the 650 m of an urban road at Manhattan (NYC), as defined by the SUMO urban mobility simulator [26], resulting in different levels of congestion over time. This configuration was based on works like [16] and [34], which simulated the performance of their proposals using different levels of vehicles density.

The mobility was implemented using the integration between the Mininet-WiFi emulator and the SUMO simulator. Therefore, the SUMO simulator was configured with the map of Manhattan roads to generate 158 vehicles' mobility, which was enough to fill the segment of road in the communication range of the RSUs. We configured three RSUs in a segment of 650 m, with the signal coverage area corresponding to 250 m in each RSU based on [35].

To observe the impact of interference caused by the vehicle's transmissions in our metrics, from the 158 vehicles, the maximum of 50 vehicles generate traffic related to different applications simultaneously, being 15 vehicles generating data of priority applications. With this scenario, it is possible to evaluate how the different approaches used can deal with topology changes, resulting in different levels of network resources consumption in a densely crowded environment, impacting the data rate KPI of the priority ITS applications. The parameters used in the performance evaluation scenario are summarized in Table II.

We have divided the vehicles' roles in a way that we have 15 vehicles that are subscribers of four hypothetical vehicular applications that are configured as described in Table III.

The application named S represents a Class A application of vehicular safety. These types of applications are regarded as vital steps towards enhancing road safety by preventing accidents from occurring [36], and is thus most critical to delays and losses. The S application has a data rate KPI of 500 Kbps, and its servers (S1, S2, and S3) are connected directly to the RAN of each RSU, following a MEC approach.

Applications E describe a Class B application of efficiency that has a data rate KPI of 500 Kbps. ITS applications for efficiency can deal with several aspects of vehicular environment (e.g., the energy efficiency of fully electric vehicles [37]). The $E2$ represents a Class C application of entertainment (e.g., a specific application of video streams), and consumes a higher bandwidth, having a data rate KPI of 1Mbps. In the evaluation scenario these applications are not delay-sensitive, and their traffic go to their remote servers through the backbone link of the RSUs. G application represents a generic traffic of 500 Kbps that has no priority, such as a generic internet browser traffic.

Each of the 15 vehicles subscriber of all applications generate the total data rate of 2 Mbps. Since the RSU backbone links are configured with a limit of 5 Mbps, these links will be congested in some periods during evaluation time, and thus the applications must be prioritized properly. The five SDN backbone switches ($SW1$ to $SW5$) interconnect the RSUs with the cloud servers of the E, $E2$ and G applications. It is worth noting that there is also an internal switch in each RSU that interconnects the RSUs with their neighboring units, the respective MEC servers, and the backbone switches.

The Mininet-WiFi does not implement 5G wireless networks. However, 5G is not about only radio access technology and encompasses several technologies in its core network [38]. Thus, we use this emulator configured with IEEE 802.11g the radio access connectivity due to its capabilities to provide realistic SDN wireless networks that are enough to deploy (in conjunction with other tools) evaluation scenarios with 5G network slicing and future ITS applications.

Based on [39], which considers as evaluation schemes the Baseline (No QoS) and Static QoS, we compare our proposed framework to two other approaches to evaluating it. One of the approaches is the same consideration is based on another

work from literature [20], where it is used SDN to manage the resources in the vehicular network considering the position of the nodes. The last approach considered deals only with the use of QoS in the Network.

These two approaches are namely the "QoS only" and "RMSDVN", as follows:

- "QoS only": The network sets QoS policy to meet the data rate KPIs for the applications. It configures QoS rules and queues, as the central SDN controller in our solution also does, but there are not local controller features, such as traffic redirection. Thus, this approach does not update the QoS according to changes in the vehicular environment.
- 'RMSDVN": The approach is implemented in [20] and uses SDN to manage the resources in the vehicular network considering the position of the nodes but does not implement traffic prioritization at all.

Finally, three metrics will be employed to evaluate the impact of the above three approaches. As [40], that use throughput to evaluate its proposal in a SDN Enabled 5G-VANET, we also use the throughput of data between vehicles and application servers. Since in our evaluated scenario the KPI of prioritized applications was the data rate, we collected the throughput over the time to verify if the application servers received the data rate expected according to vehicles transmission. The other metrics are the round trip time (RTT), and packet delivery ratio (PDR). The RTT can be defined as the time taken by an application starting from the initiator node (source vehicle) sending a message until receiving a response from the core network [41]. Knowing the value of RTT is important because it is proportional to congestion [42], and it is also possible to verify the impact in the communication flows of the S application. With PDR it is possible to evaluate the level of data loss in function of each approach used since it represents the average ratio of the number of successfully received data packets at the destination to the number of data packets sent. This metric is widely used to evaluate the performance of vehicular networks, including scenarios with multiple QoS constraints [43].

The values of RTT calculated by the "PING" ICMP messages in log files were summed each second for all vehicles and divided by the number of cars transmitting. In (1) is calculated the value of average RTT every second, where $RTTV_i$ are the $n-th$ values received of RTT in the second s and NC is the number of vehicles that transmitted in the congestion period to which the second s is part.

$$RTT(s) = \frac{1}{NC}\sum_{i=1}^{n} RTTV_i \tag{1}$$

The PDR was calculated through the ratio between the sum of packages received by servers and the number of packages sent by vehicles, as showed in (2), where PS_r is the sum of packets received by the servers and PS_t is the sum of packets sent by the cars.

$$PDR = \frac{PS_r}{PS_t} \tag{2}$$

To compute the throughput, since the data with the size of the packets sent by vehicles and received by servers collected was is in bytes, they were summed in each second and multiplied by 8 to obtain the result in bits per second (bps). The values obtained were divided by the number of vehicles that were effectively transmitting in each second, as shown in (3) and (4). $TTC(s)$ and $TRS(s)$ are, respectively, the throughput of packets sent by vehicles and received by the application servers in the second s, while PS_i and PR_i are, respectively, the n packages sent by the NC vehicles and received by the servers in the second s. NC is increasing over time when more vehicles join to the traffic jam.

$$TTC(s) = \frac{1}{NC}(\sum_{i=0}^{n} PS_i) * 8 \tag{3}$$

$$TRS(s) = \frac{1}{NC}(\sum_{i=0}^{n} PR_i) * 8 \tag{4}$$

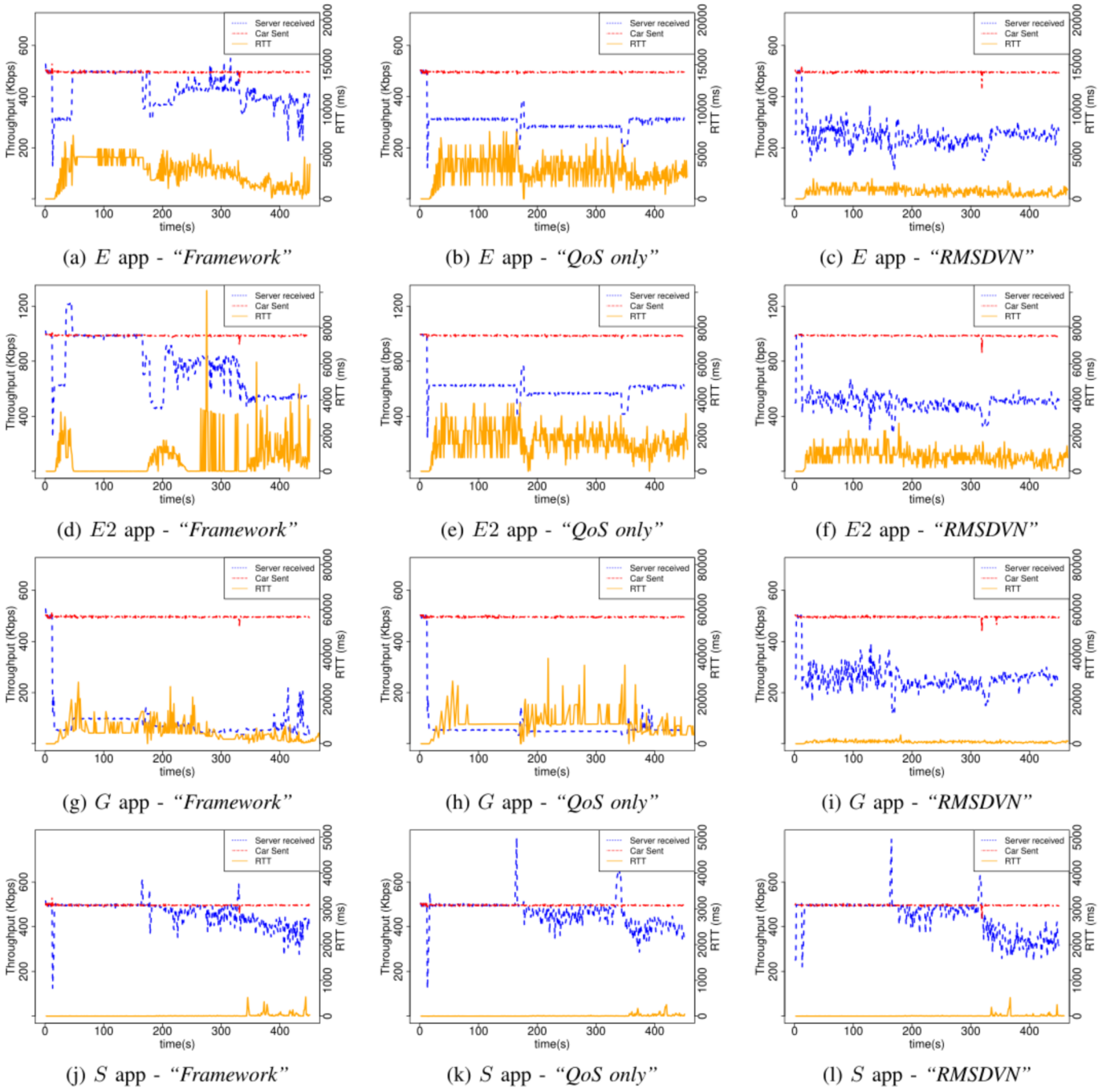
V. Results

A. Throughput and RTT Over Time

Figures 4 (a), (b), and (c) illustrate the results of throughput and RTT related to E application, using the *"Framework"*, *"QoS only"* and *"RMSDVN"* approaches, respectively. This application is the most critical among those that are prioritized in backbone links.

Figures 4 shows the results of throughput and RTT during the evaluation time for the E, $E2$, G, and S applications, using the different approaches evaluated. With all approaches, the throughput that reaches the E application server is compromised at the beginning of the evaluation time, and while each vehicle generates a throughput of around 500 Kbps for this application, in the server reaches only around 300 Kbps. This occurs because there are until 48 vehicles reaching RSU3, 17 generating several data rates, with being 5 subscribers of the applications E and $E2$ among them. Since each one of these 5 vehicles generate 2 Mbps throughput to backbone, being 1 Mbps for $E2$ server, 500 Kbps for E server, and 500 Kbps for G server, totalizing 10 Mbps requirement for the 5 Mbps uplink of the RSU. There are also 500 Kbps to S application in local MEC server but this is not considered by the controller, since these flows do not use the uplink of the RAN. With our proposal, the local controller redirected 3 Mbps related to communication flows of $E2$ application from three vehicles, and the QoS limited the G application. Thus, it is possible to observe that after a convergence time, there is almost no difference between the throughput generated by vehicle and received by servers until around 200s. Around this time, there are 34 vehicles generating data in RSU1 and RSU2, with 138 vehicles in the traffic jam. The density of vehicles increases in order that during the last 100s there is the maximum level of traffic jam in the experiments, where there are 158 vehicles, with 50 generating data in the RANs, being 15 of these vehicles subscribers of priority application.

In summary, with the proposed framework were obtained the best results regarding data rate overtime to E application. Even with the impact observed in the results related to RTT, because of the QoS, our approach presented better results for this metric than using *"QoS only"* approach. Related to



(a) E app - *"Framework"* (b) E app - *"QoS only"* (c) E app - *"RMSDVN"*

(d) $E2$ app - *"Framework"* (e) $E2$ app - *"QoS only"* (f) $E2$ app - *"RMSDVN"*

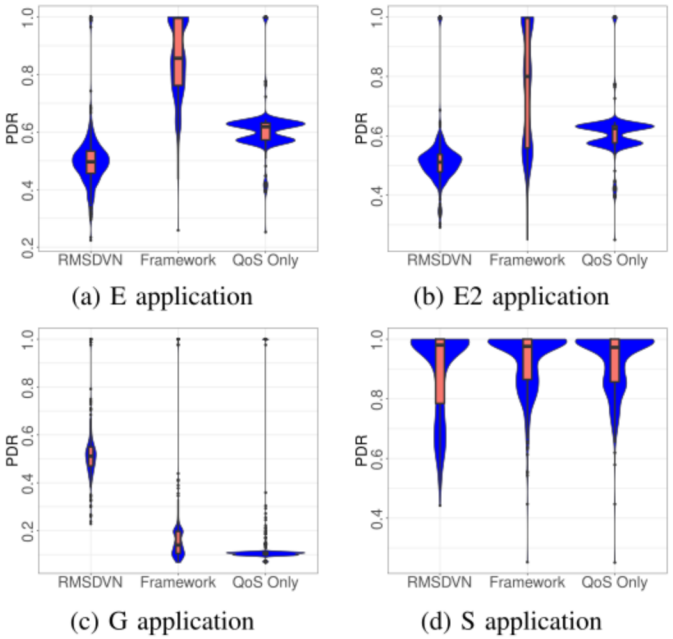
(g) G app - *"Framework"* (h) G app - *"QoS only"* (i) G app - *"RMSDVN"*

(j) S app - *"Framework"* (k) S app - *"QoS only"* (l) S app - *"RMSDVN"*

Fig. 4. Results of throughput and RTT during the evaluation time for the E, $E2$, G, and S applications, using the different approaches evaluated.

the $E2$ application, the results obtained with the proposed framework were similar to that using *"QoS only"* approach, with the worst results in the last 100s, as expected, since the solution prioritized the E application. RTT values were proportional to the congestion level and naturally worst when the traffic was limited. Since *"RMSDVN"* does not inspect the traffic to provide some prioritization, its RTT values are lower. In any case, some high RTT values are not a problem in the evaluated scenarios since the KPI used as a reference was the data rate. In respect of G application, it is possible to observe the best results with the *"RMSDVN"* approach, and this was expected since both the proposed framework and the *"QoS only"* approaches limit the communication of this application to prioritize the others.

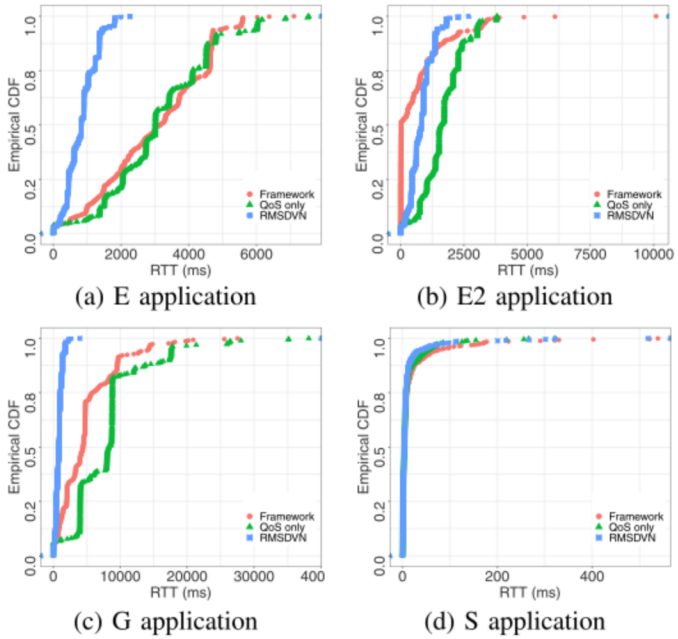
Figures 4 (j), (k), and (l) show the RTT results, transmission and reception throughput rates collected over time for the S application. Since the data of these applications do not pass through the network backbone, the expected result is that it be independent of the approach used. The results are very similar, which are consistent with the theory, except for some outliers. It is also observed that, although congestion in the backbone does not impact the traffic of these applications, it is



(a) E application (b) E2 application

(c) G application (d) S application

Fig. 5. Packet delivery ratio for the different applications on the evaluated approaches.



(a) E application (b) E2 application

(c) G application (d) S application

Fig. 6. ECDFs of the RTT results for the different applications using the evaluated approaches.

not the immune to the high interference of the 50 vehicles generating different data rates in the RANs. Therefore, it is possible to observe a decrease in reception rate during the last 100s of evaluation time in all figures, in function of high density of vehicles transmitting. The RTT values were shallow, as expected in function of the use of MEC severs.

B. Packet Delivery Ratio

In this section we analyze the PDR values obtained according to the use of the evaluated approaches. Figure 5 (a), (b), (c), and (d) show the violin plot with the PDR calculated during the experiments using the evaluated approaches for the E, $E2$, G, and S applications, respectively.

Violin plots with the PDR values observed during the experiments with the evaluated approaches for the different applications are shown in Figure 5. It is possible to observe that the values using *"RMSDVN"* approach mainly focus around 40 and 60% for E, $E2$, and G applications since there is no prioritization in backbone when using this approach. With *"QoS only"* the results are similar for the priority applications E and $E2$, since, with the QoS rules and queues, it is possible to allocate some network bandwidth. Despite that, there is no traffic redirection in this approach. With the packet losses, the PDR values with *"QoS only"* were focused around 60%, except for the G application, which was penalized, once there is no priority. Our solution obtained the best PDR values for E and $E2$ applications, which occurs because our framework considers the application requirements and mobility in each RAN. Even for the G application, which is penalized, our proposal presented better results than with *"QoS only"* approach.

In summary, considering all the evaluation time, the mean PDR with our proposal was about 85.1% for E application against respectively 50.1% and 60.8% with *"RMSDVN"* and *"QoS only"*. With $E2$ application the mean PDR was 76.9% with our proposal, against respectively 51.1% and 61.1% with *"RMSDVN"* and *"QoS only"*. These results make clear the improved performance of our proposal when considering the PDR metric to evaluate the operation related to the priority applications.

Considering the values of all emulation time, the mean PDR with all approaches in S application is around 90%, which confirms the benefits of use MEC solution and the independence of this from the approach used. These results are practically equal since the data from the S application does not go through the network core and, therefore, is not manipulated by the implemented approaches. On the other hand, the observations of PDR lower than 90% are more evidence of the impact of the interference generated by vehicles in each RAN during the periods with a higher density of vehicles.

C. Round Trip Time

The results of RTT through empirical cumulative density functions (ECDFs) for E, $E2$, G, and S applications are illustrated in Figures 6 (a), (b), (c), and (d), respectively.

The ECDF of E application shows that more than 90% of the values are less than 1750 ms with the *"RMSDVN"* approach, while with the our solution and *"QoS only"* are less than around 5000 ms. As the use of QoS affects the RTT, the best results are with the *"RMSDVN"* approach when considering the applications that use the uplinks in the RANs.

It was possible to observe that, once our framework use QoS applied by the central controller, the RTT values were high than with *"RMSDVN"*. But, this is not a problem in this scenario, since, as mentioned before, this was not the KPI considered by the control algorithms. The S application is that with lower RTT values.

For the *S* application, the RTT measurements in the data communications between the vehicles and the MEC servers show excellent and very similar results of low RTT with the evaluated approaches, with a mean of around 20 ms, independent of the used approach. This shows the benefits of MEC servers concerning this metric. Other conclusion based on these results is that the high interference does not impact the RTT in the same level of the PDR metric.

D. Discussion

The results showed that with the use of the proposed framework it is possible to orchestrate the network resources to respond to the dynamics of the vehicular environment to meet the ITS applications' priorities and its KPIs. Compared with the other approaches, it was possible to obtain promising results of reception data rate in application servers, and PDR, in extreme conditions of data congestion and interference in a vehicular scenario with varying network topology over time.

It was observed in the results that the use of QoS policies could increase the RTT. Since the applications sensitive to time constraints could use a MEC approach, and in the scenario of our proof-of-concept, the KPI used by the controllers to manage the applications in conjunction with its priority was the data rate.

The proposed solution presented for the *E* application a PDR result of 39.96% above that achieved with *"QoS only"* approach and 69.86% above that achieved with *"RMSDVN"* approach. It is noteworthy that all results could be even better if there was not the limitation in the local controller implementation, where, it was necessary to wait at least 20 seconds to act after detecting the need to intervene to meet the applications' KPIs, which is a significant period of time, considering the total evaluation time of 450 seconds.

VI. Conclusion

This work addressed the problem of how to deploy a mobile infrastructure with a vehicular network that can dynamically meet the communication requirements of different ITS applications. The proposed approach consists of a framework using 5G network slicing concepts in a software-defined vehicular network with algorithms that consider the requirements of the different applications, its infrastructure, and the vehicles' mobility, to meet applications requirements.

Throughout the realistic experimental evaluation, we were able to validate the approach and show how the data rate received by application servers over time, RTT, and PDR of the communication flows were used as performance metrics. The obtained results were outperforms alternative approaches such as *"QoS only"* and *"RMSDVN"* in scenarios of dynamic traffic jam with vehicles associated to four applications with different data rate requirements and priorities.

References

[1] I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. Imran, and N. Guizani, "Overcoming the key challenges to establishing vehicular communication: Is SDN the answer?" *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 128–134, Jul. 2017.

[2] S. A. A. Shah, E. Ahmed, M. Imran, and S. Zeadally, "5G for vehicular communications," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 111–117, Jan. 2018.

[3] R. Molina-Masegosa and J. Gozalvez, "LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications," *IEEE Veh. Technol. Mag.*, vol. 12, no. 4, pp. 30–39, Dec. 2017.

[4] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive Internet of vehicles," *Comput. Commun.*, vol. 120, pp. 58–70, May 2018.

[5] X. Li *et al.*, "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, Sep. 2017.

[6] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[7] C. Campolo, R. Fontes, A. Molinaro, C. E. Rothenberg, and A. Iera, "Slicing on the road: Enabling the automotive vertical through 5G network softwarization," *Sensors*, vol. 18, no. 12, p. 4435, Dec. 2018.

[8] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69, Mar. 2008.

[9] L. Zhang and S. Valaee, "Congestion control for vehicular networks with safety-awareness," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3290–3299, Dec. 2016.

[10] ETSI. (2017). *Network Functions Virtualisation (NFV); Management and Orchestration*. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[11] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[12] *OpenBaton*. Accessed: Oct. 9, 2020. [Online]. Available: https://openbaton-docs.readthedocs.io/en/latest/

[13] C. Huang, R. Lu, and K.-K.-R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[14] M. Tao, J. Li, J. Zhang, X. Hong, and C. Qu, "Vehicular data cloud platform with 5G support: Architecture, services, and challenges," in *Proc. 7 IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 32–37.

[15] S. Correia, A. Boukerche, and R. I. Meneguette, "An architecture for hierarchical software-defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 80–86, Jul. 2017.

[16] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[17] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei, "Soft-defined heterogeneous vehicular network: Architecture and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 72–80, Jul. 2016.

[18] A. Kaul, K. Obraczka, M. A. S. Santos, C. E. Rothenberg, and T. Turletti, "Dynamically distributed network control for message dissemination in ITS," in *Proc. IEEE/ACM 21st Int. Symp. Distrib. Simulation Real Time Appl. (DS-RT)*, Oct. 2017, pp. 1–9.

[19] F. S. Tegueu, S. Abdellatif, T. Villemur, P. Berthou, and T. Plesse, "Towards application driven networking," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Jun. 2016, pp. 1–6.

[20] R. Dos Reis Fontes, C. Campolo, C. E. Rothenberg, and A. Molinaro, "From theory to experimental evaluation: Resource management in software-defined vehicular networks," *IEEE Access*, vol. 5, pp. 3069–3076, 2017.

[21] 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.

[22] A. A. Khan, M. Abolhasan, and W. Ni, "5G next generation VANETs using SDN and fog computing framework," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2018, pp. 1–6.

[23] D. A. Chekired, M. A. Togou, L. Khoukhi, and A. Ksentini, "5G-slicing-enabled scalable SDN core network: Toward an ultra-low latency of autonomous driving service," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1769–1782, Aug. 2019.

[24] V. Balasubramanian, S. Otoum, M. Aloqaily, I. Al Ridhawi, and Y. Jararweh, "Low-latency vehicular edge: A vehicular infrastructure model for 5G," *Simul. Model. Pract. Theory*, vol. 98, Jan. 2020, Art. no. 101968, doi: 10.1016/j.simpat.2019.101968.

[25] I. A. Ridhawi, S. Otoum, M. Aloqaily, and A. Boukerche, "Generalizing AI: Challenges and opportunities for plug and play AI solutions," *IEEE Netw.*, vol. 35, no. 1, pp. 372–379, Jan. 2021, doi: 10.1109/MNET.011.2000371.
[26] D. Krajzewicz, J. Erdmann, M. Behrisch, and B. Laura, "Recent development and applications of SUMO-simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, nos. 3–4, pp. 128–138, 2012.
[27] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervas. Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.
[28] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted over-the-top edge prefetching for hierarchical VANETs," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1786–1801, Aug. 2018.
[29] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should I slice my network?: A multi-service empirical evaluation of resource sharing efficiency," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2018, pp. 191–206.
[30] Open Network Foundation. (Feb. 2016). *SDN Architecture 1.1*. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf
[31] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "MiniNet-WiFi: Emulating software-defined wireless networks," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 384–389.
[32] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (Hotnets)*, Oct. 2010, pp. 1–6.
[33] R. Cziva, S. Jouet, D. Stapleton, F. P. Tso, and D. P. Pezaros, "SDN-based virtual machine management for cloud data centers," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 2, pp. 212–225, Jun. 2016.
[34] S. Kuhlmorgen, H. Lu, A. Festag, J. Kenney, S. Gemsheim, and G. Fettweis, "Evaluation of congestion-enabled forwarding with mixed data traffic in vehicular communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 233–247, Jan. 2020.
[35] H. Li, M. Dong, and K. Ota, "Control plane optimization in software-defined vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7895–7904, Oct. 2016.
[36] S. Al-Sultan, A. H. Al-Bayatti, and H. Zedan, "Context-aware driver behavior detection system in intelligent transportation systems," *IEEE Trans. Veh. Technol.*, vol. 62, no. 9, pp. 4264–4275, Nov. 2013.
[37] Y. Wang, J. Jiang, and T. Mu, "Context-aware and energy-driven route optimization for fully electric vehicles via crowdsourcing," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1331–1345, Sep. 2013.
[38] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.
[39] E. Ahmad, M. Alaslani, F. R. Dogar, and B. Shihada, "Location-aware, context-driven QoS for IoT applications," *IEEE Syst. J.*, vol. 14, no. 1, pp. 232–243, Mar. 2020.
[40] X. Duan, Y. Liu, and X. Wang, "SDN enabled 5G-VANET: Adaptive vehicle clustering and beamformed transmission for aggregated traffic," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 120–127, Jul. 2017.
[41] A. M. Said, M. Marot, A. W. Ibrahim, and H. Afifi, "Modeling interactive real-time applications in VANETs with performance evaluation," *Comput. Netw.*, vol. 104, pp. 66–78, Jul. 2016.
[42] P. K. Sahoo and Y. Yunhasnawa, "Ferrying vehicular data in cloud through software defined networking," in *Proc. IEEE 12th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2016, pp. 1–8.
[43] M. H. Eiza, T. Owens, Q. Ni, and Q. Shi, "Situation-aware QoS routing algorithm for vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5520–5535, Dec. 2015.

Tiago do Vale Saraiva received the M.Sc. degree in informatics from the Federal University of the State of Rio de Janeiro (UNIRIO), Brazil, in 2018, with the dissertation on an application-driven framework using software-defined networks and 5G network slicing for intelligent transportation systems, where he is currently pursuing the D.Sc. degree in informatics. Since 2010, he has been with the Petróleo Brasileiro S. A. (PETROBRAS), a Brazilian oil company. His experience and research interests include cybersecurity, machine learning, intelligent transportation systems, mobile computing, software-defined networks, and the Internet of Things.

Carlos Alberto Vieira Campos (Member, IEEE) received the B.Sc. degree in computer science from the State University of Montes Claros, Brazil, in 2000, and the M.Sc. and D.Sc. degrees in systems engineering and computer science from the Federal University of the State of Rio de Janeiro, Brazil, in 2003 and 2009, respectively. From 2018 to 2019, he was a Visiting Professor with the Department of Electrical and Computer Engineering, University of Toronto, Canada. He is currently an Associate Professor with the Federal University of the State of Rio de Janeiro. His current research interests include user mobility modeling, wireless networks, vehicular networks, mobile computing, and intelligent transportation systems. He served as a Technical Committee Member for the VTC, WCNC, ISCC, ICCVE, and others conferences. In addition, he served as a reviewer for several journals and magazines.

Ramon dos Reis Fontes is currently a Professor in computer science with the Federal University of Rio Grande do Norte (UFRN), Natal, Brazil. Before his current appointment he received the M.Sc. degree from the University of Salvador (UNIFACS), Salvador, Brazil, in 2014, and the Ph.D. degree in electrical and computer engineering from the State University of Campinas, Brazil, in 2018. During his Ph.D., he developed a wireless network emulator that has been widely used by researchers worldwide, and was a Visiting Ph.D. Student with the DIANA Team, Sophia Antipolis, France, in 2016. From June 2011 to July 2011, he was a Professor with the Federal Institute of Science and Technology of Bahia (IFBA), where he taught subjects related to computer networks, including network operating systems, routers, application servers, information security, and programming. His main research interests are in the field of software defined wireless networks (SDWN), vehicular networking, future Internet architecture, and cyber security.

Christian Esteve Rothenberg received the degree in telecommunication engineering from the Technical University of Madrid (ETSIT–UPM), Madrid, Spain, in 2006, the M.Sc. (Dipl.Ing.) degree in electrical engineering and information technology from the Technical University of Darmstadt, Germany, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Campinas (UNICAMP), Brazil, in 2010. From 2010 to 2013, he worked as a Senior Research Scientist in the areas of IP systems and networking, leading SDN research with the CPQD Research and Development Center in Telecommunications, Campinas, Brazil. He is currently an Assistant Professor (tenure-track) and the Head of the Information & Networking Technologies Research & Innovation Group (INTRIG), School of Electrical and Computer Engineering (FEEC), UNICAMP. His research activities span all layers of distributed systems and network architectures and are often carried in collaboration with academia and industry (e.g., Ericsson, Samsung, CPQD, Padtec, RNP) around the world, leading to multiple open source networking projects (e.g., RouteFlow, libfluid, ofsoftswitch13, Mininet-WiFi) in the areas of SDN and NFV, among other scientific results. He has contributed to five international patents, coauthored two books, and over 150 publications, including scientific journals, altogether featuring over 8000 citations. He was an Open Networking Foundation (ONF) Research Associate from 2013 to 2017, the Co-Chair of the IEEE SDN Outreach Committee Initiative from 2016 to 2017, and the Brazilian Coordinator of the EU-Brazil H2020 NECOS (Novel Enablers for Cloud Slicing) Project from 2017 to 2020. He is a member of the CPQD Innovation Committee (2017–2021) and Outreach Coordinator of FEEC/UNICAMP (2019–2023).

Sameh Sorour (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from Alexandria University in 2002 and 2006, respectively, and the Ph.D. degree from the University of Toronto in 2011. He is currently an Assistant Professor with Queen's University, Canada. His Ph.D. thesis was nominated for the Governor General's Gold Medal Award. After his graduation, he held a MITACS Industrial Post-Doctoral Fellowship with Siradel Canada and the University of Toronto. Prior to moving to Queen's University in 2019, he held another Post-Doctoral Fellowship with the King Abdullah University of Science and Technology (KAUST), and an Assistant Professor with the King Fahd University of Petroleum and Minerals (KFUPM) and the University of Idaho. During his Ph.D. and postdoctoral fellowships, he led several research projects with industrial partners and government agencies, such as LG Korea, the European Space Agency, the Canadian National Institute for the Blind (CNIB), and Siradel France. He is an Editor of IEEE COMMUNICATIONS LETTERS. His research interests include advanced computing, learning, networking technologies for cyber-physical and autonomous systems, cloud/edge/IoT computing, learning, networking, and their applications in multimodal/coordinated autonomous driving, autonomous/electric mobility on-demand systems, layered/virtualized management of future transportation networks, tactile cyber-physical systems, and smart energy and healthcare systems.

Shahrokh Valaee (Fellow, IEEE) is currently a Professor with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto. He is also the Founder and the Director of the Wireless and Internet Research Laboratory (WIRLab), University of Toronto. He is a fellow of the Engineering Institute of Canada. He was the TPC Co-Chair and the Local Organization Chair of the IEEE Personal Mobile Indoor Radio Communication (PIMRC) Symposium 2011. He was the TCP Chair of PIMRC2017, the Track Co-Chair of WCNC 2014, and the TPC Co-Chair of ICT 2015. He has been the guest editor for various journals. He is a Track Co-Chair for PIMRC 2020 and VTC Fall 2020. From December 2010 to December 2012, he was the Associate Editor of the IEEE SIGNAL PROCESSING LETTERS. From 2010 to 2015, he served as an Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He is an Editor of *Journal of Computer and System Science*.