Dynamic Controller Instantiation in MEC-NFV based Software Defined Internet of Vehicles

1st Atricia Sabino
Center of Informatics (CIn)
Federal University of Pernambuco
Recife, Brazil
masm2@cin.ufpe.br

2nd Rhodney Simoes
Center of Informatics (CIn)
Federal University of Pernambuco
Recife, Brazil
rambks2@cin.ufpe.br

3rd Kelvin Lopes Dias
Center of Informatics (CIn
Federal University of Pernambuco
Recife, Brazil
kld@cin.ufpe.br

***Abstract*—Softwarization of vehicular networks has been a fundamental approach to deal with its dynamic topology along with requirements of scalability and flexible resource management in order to deploy future Internet of Vehicles (IoV). Despite Software-defined Vehicular Networks (SDVN) has gained momentum in industry and academia to tackle the aforementioned issues, some requirements such as on-demand allocation of network functions and scalability support are not yet satisfactorily addressed in current proposals. Even scalable solutions based on distributed and hierarchical SDN controllers are commonly deployed statically before the network operation. With the advent of network functions virtualization (NFV), it is expected that IoV will benefit from a standard architecture for dynamic instantiating of SDN controllers to provide an efficient solution for IoV management. This article proposes and evaluates a flexible and dynamic solution for the management of IoV through the synergy between SDN and NFV paradigms in order to provide on-demand instantiation of SDN controllers, as well as meet QoS requirements. The results demonstrate the effectiveness of the proposal in supporting the increasing demand for connected vehicles, reduction in packet loss, jitter control, and avoiding processing overhead when compared to the performance of traditional architecture.**

***Index Terms*—SDN, NFV, MEC, VANETs, IoV**

I. Introduction

With the advent of the Internet of Vehicles (IoV), innovative services can be enabled to improve passenger comfort, accident prevention, and enhanced mobility in large urban centers [10]. In Vehicle-to-Infrastructure (V2I) Communication, the vehicle exchanges information with the wireless access points named Road Side Units (RSUs) [13]. Regarding the current architecture, some challenges in IoV require more research and new solutions to tackle the significant increase of connected devices, the need for efficient usage of resources, dynamic nature of vehicular data traffic, delays, and packet losses due to the high mobility of vehicles, unreliable connections, and Quality of Service (QoS) requirements of the applications. To address the challenges of managing IoV, authors in [5] proposed the adoption of Software-Defined Networking (SDN).

The SDN [4] paradigm is based on the separation of data and control planes, as well as a global network view to provide flexible and programmable management solutions. Thus, the so called Software Defined Vehicular Networks (SDVN) provides a flexible control plane for managing the communication between RSUs and vehicles [1]. However, despite its distributed control, most implementations still use a single SDN controller per domain as in [3] or even designs that deploy multiple controllers based on static and offline approaches [8].

This can lead to problems of scalability and failures, especially in a dynamic environment such as the IoV scenarios. SDVN architectures allow the controller to manage communication between vehicles and RSUs. During the network operation, several messages are exchanged between the controller and the vehicular data plane. The amount of flows due to the signaling operations between vehicles and RSUs increases according to the number of new vehicles entering the network or existing ones making handovers among RSUs or even to assist network policies and users demands, that will issue flow rules from SDN controllers to populate vehicles flow tables. This may overload the SDN controller and degrade the network performance and its QoS guarantees, which causes the high number of new requests to congest the SDN controller.

To deal with such a problem, while still meeting the latency constraints of IoV environment applications [12], it is necessary to adopt an infrastructure apt to provide dynamic, scalable, and low latency capabilities for the deployment of SDN controllers. Therefore, the joint Multi-access Edge Computing (MEC) and Network Functions Virtualization (NFV) environment can provide such benefits for IoV demands [14].

This article proposes and evaluates a flexible and dynamic solution for the management of vehicular networks through the synergy between SDN and MEC-NFV environment. The proposal devises an architecture and a strategy of auto-scaling of SDN controllers configured as Mobile Edge Applications (ME apps) according to the traffic demands in vehicular networks. The proposal is implemented in a MEC-NFV testbed using the tacker service based on the NFV MANO architecture

integrated with the Openstack platform. The network scenarios use the wireless SDN network emulator called Mininet-WiFi [3], integrated into the Simulation of Urban Mobility (SUMO). The results obtained demonstrate the effectiveness of the proposed solution to support the increasing demand of connected vehicles, while reducing packet losses, controlling jitter, and avoiding processing overhead when compared to the performance of traditional architecture.

This article is organized as follows: Section II highlights and discusses the related work. Section III presents the proposed integrated. Section IV discusses the results obtained by a set of experiments carried out considering scenarios and metrics. Finally, V concludes the work summarizing the results achieved and future works.

II. Related Work

SDN and NFV have been adopted by vehicular networks in order to guarantee flexibility, programmability, cost reduction of both deployment and operation, and scalability. This section presents some related works applying those softwarization paradigms either individually or cooperatively in the contexts of VANETs and Internet of vehicles (IoV).

In [5], a single centralized SDN controller (control plane), deployed and accessed via cellular connectivity (3G/4G), is in charge of the entire VANET, while user data is transmitted through RSUs (IEEE 802.11) in both V2V and V2I modes. Despite also considering a distributed approach to switch directly to V2V communication mode in case of cellular connectivity loss with the control plane, that is, fault of the centralized SDN controller, this solution relies on using traditional MANETs routing protocol on the data path, thus lacking the benefits of SDN. The authors compare their approach against traditional MANET routing protocols through NS-3 simulations, which demonstrate the packet rate delivery (PDR) gains and the effectiveness of the fallback mechanism.

A hierarchical SDVN architecture is proposed in [2]. Two layers: local SDN controllers at the bottom and a single main controller at the top level of the hierarchy. The clustering technique is adopted in order to choose a cluster head in charge of becoming the SDN controller (either an RSU or a vehicle) for the corresponding domain. Local SDN controllers may serve vehicles during connectivity loss due to the failure of the main controller. Despite the dynamic assignment of SDN controllers to RSU or vehicles, both nodes should be SDN-aware, thus the solution does not contemplate legacy VANETs nodes. Furthermore, deployment of SDN controllers might not be arranged in a flexible manner and within bounded delays, since it requires running complex clustering algorithms in real-time, in contrast to the easy on-demand deployment of virtual functions over the cloud computing platform.

The proposal of [3] creates and implements a vehicles architecture for VANETs, named cars as node (node car), adapted for the Mininet-WiFi network emulator. The tool has support for the OpenFlow protocol based on lightweight virtualization, which adds to the emulator the function of wireless channel emulation and the mobility support. The technique covered in [3] installs the static flows and modifies them in the wired aggregation switch (each switch has its ports connected to different wireless access points) to emulate the mobility of cars in a controlled manner. The evaluation scenario take into account a video streaming application. During the experiment, execution, when the car with ongoing video transmission leaves its current BS coverage, the SDN controller will detect this event and coordinate a new connection to an RSU or another BS.

In [8], the SDN controller is positioned at RSUs in order to reduce the operational latency for vanet communications. Subsequently, this previous work is extended with a hierarchical SDVN architecture in [6] with two tiers: at the top level, that is, the Internet level, there are distributed regional controllers and at the bottom level, selected RSUs running local controllers. A heuristic model of controller positioning is used at the RSU level. Despite the benefits of the proposal in terms of scalability and latency, since it has to select a few RSUs to host local controllers due to a high cost to deploy SDN controllers for all RSUs of the topology, it is not flexible or of easy management. Besides that, its correct operation also may depend on cellular/4G interfaces on OBUs for accessing regional controllers to benefit from the global view of the network.

To overcome shortcomings during the handover process for time-sensitive applications, the benefits of fog computing, network virtualization capabilities, and SDN programming are used to create an architecture that allows planning handovers with travel information and providing scalable solutions [7]. In [9], a fog computing solution is proposed for supporting vehicular applications and decision making. The SDN / NFV and fog are used to create an architecture capable of remotely controlling connected cars through the data modeling language YANG (Yet Another Next Generation). The Fog proposal is based on Docker and OpenVSwitch with containers for each service that is needed.

Our proposal is distinguished from related work since it devises a scalable SDVN architecture to dynamically deploy SDN controllers while granting low latency applications to serviced cars. Therefore, the proposal encompasses MEC in an NFV environment and located at the eNBs. Where the controllers are dynamically allocated as MEC apps depending on the demand. To validate the proposal, the IoV scenario was emulated to generate signaling and traffic load to the MEC testbed implemented in an NFV enabled Openstack infrastructure.

III. MEC architecture in an NFV environment for SDVN

In this article, we propose to deploy dynamically SDN controllers in a MEC architecture for IoV called MEC-SDIoV, aiming at reducing the communication latency between controllers and RSUs without compromising the scalability of the environment. In addition, our solution provides joint programmability and flexibility advantages to dynamic IoV

scenarios by benefiting from distinguished features of SDN, NFV, and, MEC.

A. *MEC-SDIoV architecture*

Vehicles in the IoV scenario are responsible for requesting contents and applications, among other demands, that generate network traffic to RSUs when using 802.11 interfaces. This communication is prioritized due to the high cost of LTE, thus RSUs are responsible for providing communication between vehicles and the Internet. The decisions on traffic engineering, load balancing, and mobility issues are managed by applications running on SDN controllers, which are dynamically deployed in the MEC-NFV architecture. This virtualized environment is located closer to the vehicles, that is, at the eNBs. Therefore, upon arrival of new packets at RSUs to which there are no flow rules already established to forward them, the RSUs generate PACKET-IN messages toward the controllers in order to acquire the required rules. Thus, as there is an increase in the number of vehicles within the coverage area of RSUs, there may be a proportional increase in demand for RSUs and, consequently, for SDN controllers. In view of the scenario described above, MEC-SDVN architecture, see Figure 1, aims at scaling dynamically SDN controllers in the MEC in an NFV environment in order to tackle the demand in terms of increasing number of control flows, that is, PACKET-IN messages. Taking a dynamic approach to scaling controllers implies the task of reassigning controllers to RSUs for load balancing among them.

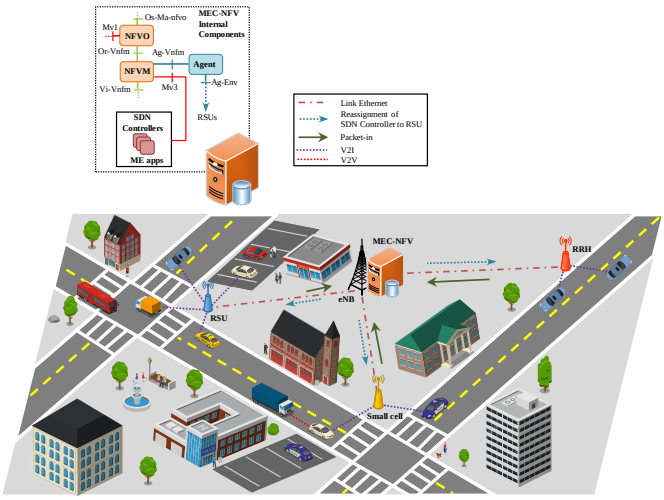

Fig. 1. MEC-SDIoV - An MEC based SDVN architecture for IoV

Our MEC-SDIoV architecture proposes an extension to the MEC reference architecture in NFV environment and its internal components based on functional blocks can be seen in Figure 1. This proposal aims to scale SDN controllers as ME apps. An Agent is added to the MEC-NFV reference architecture, which is responsible for periodically requesting information from the Virtual Network Functions Manager (VNFM) with regard to the allocation of new controllers through the reference point Ag-Vnfm. Upon a trigger event to create new controlers, the Agent decides on the Controller Reassignment-RSUs task for load balancing between the controllers and then applies the decision made through the AgEnv RSUs reference point.

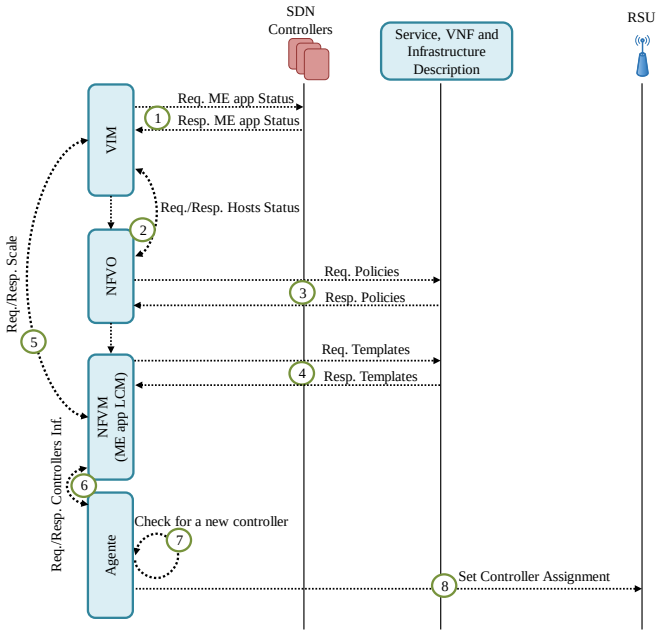

Fig. 2. Proposal Signaling

The numbers depicted in Figure 2 summarize the interaction components: In step **(1)**, VIM performed the monitoring of controllers instantiated as ME apps. Next,the network functions virtualization orchestration queries the VIM about the infrastructure resources in step **(2)**. Then, in the step **(3)** the NFVO queries defined policies and sends it to the VNF manager. In the step **(4)** the VNF manager queries which templates are defined in the VNF descriptor. Following, in step **(5)** auto-scaling actions are performed. **(6)** the agent monitors VNFs through VNF manager. **(7)** Check if a new SDN controller is deployed. Finally, in **(8)** the agent sends commands to reassign the RSUs.

IV. Testbed Overview and Proposal Evaluation

A. *Testbed*

In order to validate MEC-SDIoV architecture, a testbed integrated with an emulation environment was implemented. In order to create a scalable IoV environment with RSUs, Mininet-WiFi in conjunction with SUMO was used. For the MEC-NFV environment, the OpenStack was integrated with NFV based on the ETSI MANO architecture through the Tacker service, which has API responsible for the VNFs management cycle, being able to monitor VNFs, perform auto scaling, and self-recovery. The Agent was implemented in Python by interacting with the Ryu controllers, as Me apps, through Tacker service and applying the reassignment decisions to the RSUs in the Mininet-WiFi.

B. Results

In this section, we present the evaluation of the proposed MEC-SDIoV architecture. Note that all experiments were performed 30 times and the graphs were generated with a 95% confidence interval.

For the sake of testing the proposal to demonstrate the operation of the architecture and preliminary performance results, it was defined as a simple policy for the instantiation of SDN controllers based on the CPU usage of ME app. One of the strategies carried out to evaluate VNFs overload is to measure the percentage of CPU usage, which is quite sensitive to large numbers of requests and data exchanges, therefore, it is widely considered in the MEC-NFV environment to assess the state and performance of ME apps. In addition, VNF overloading may result in fewer packets being processed, leading to packet losses and delays. To identify the impact on packet loss and throughput, an experiment to evaluate the QoS impact of a file transfer application was performed. For this purpose, the application traffic was generated via Iperf tool. The download rate was set to 20Mbps. In this way, it was possible to simulate the download or an information transfer between a requesting car and the file server in the MEC. Each RSU is connected to a remote SDN controller. Simulation time was set to 60s. The average value of metrics were plotted considering three different number of vehicles (120, 260, 310) and 4 RSUs. The evaluation of the proposal was conducted in an MEC testbed integrated with a vehicular network emulator. Four experiments were conducted, ranging from a scenario with a single controller to four dynamically instantiated controllers. Thus, even under simple scaling policies, the proposal provides a flexible way to meet scalability. To define the threshold of the CPU utilization to trigger the auto scaling policy, simulations were performed to monitor the VNF behavior under increasing traffic scenarios. It was possible to identify that at the moment in which the processing load reaches 60% of CPU capacity, losses of packets begin to occur.

C. Results and discussion

The experiments and analysis of the results take into account the metrics CPU usage, packet loss, and jitter. The first experiment is depicted in Figure 3, which shows the CPU utilization over time considering three different amount of vehicles in order to test the auto scaling feature of our proposal. The three evaluation scenarios initially start with a single SDN controller to meet the vehicles requirements. The CPU utilization is constantly monitored by the architecture, besides the checking for the need to allocate a new SDN controller. Then, the NFVO deploys another VNF based on predefined threshold. Thus, it is possible to observe in Figure 3 such behavior and how the demand from the vehicles is balanced among the controllers over time. The SDN controllers instantiated as VNFs are represented in the legend as C1 for the case of a single controller experiment, C2 for the instantiation of the second controller, C3 for the case of three controllers, and the fourth controller instantiation is represented by C4.

The graph of Figure 3 (a) depicts the average CPU usage over time for the single controller dynamically instantiated. When the first VNF (C1) reached the threshold of 60% regarding CPU usage, a new VNF (C2) is instantiated. The percentage of CPU utilization of the second controller represented in Figure 3(a) as C2 is shown only 10 seconds after starting its instatiation. This is because until that time the corresponding Ryu controller had not yet become fully operational. Despite this instantiation time, the adopted threshold of 60% provides a safe margin for minimum packet losses for these operations.

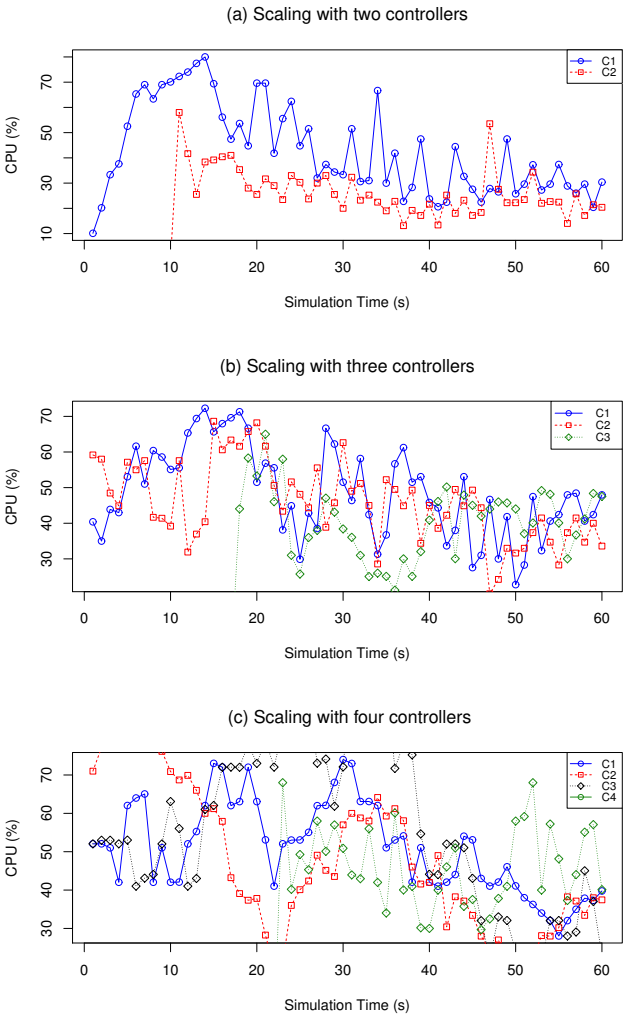

Fig. 3. Dynamic scaling of SDN controllers as VNFs

For the next experiment depicted in Figure 3(b), the scenario encompasses three controllers, where two of them are overloaded so that the action of instantiating a new VNF is triggered. The parameters in the Mininet-WiFi are also modified because more cars and information exchange are required to send as many PACKET-IN messages as possible in order to increase the load on the controllers. The number of 260 vehicles and 4 RSUs was sufficient for the two controllers to reach the threshold of 60% regarding CPU utilization. From the analysis of results depicted in Figure 3 (b), despite reaching the 60% threshold in the first few seconds of the simulation, the scaling action on Tacker NFV is not performed. The execution of the task occurs only within the next 20 seconds. Regarding the scaling shown in the previous subsection, VNF lasts more time congested, that is, the time to trigger a new instance is greater due to the use of ceilometer monitoring overhead, which now has one more VNF to collect data and also requires more MEC resources. The third controller is created to run in parallel with the two already existing ones. And the whole process for balancing traffic among RSUs occurs again. Finally, in the last scenario depicted in Figure 3 (c), the goal is to report the case for a greater instantiation of SDN controllers as VNFs. The load generation for the IoV scenario has now been configured with 310 cars. The intention is to reproduce the behavior of city with a larger number of cars, justified, for example, in large events: such as the World Cup or Olympics game. With this increased vehicular flow and countless amounts of message exchanges, more SDN controllers need to be deployed. Figure 3 (c) shows that for this experiment, only three SDN controllers are running before 20 seconds. The operation that creates new scaling tasks has a setting to monitor the CPU percentages of the three controllers. When any of the VNFs reaches 60% a new one is created. This behavior has been configured in templates TOSCA (Topology and Orchestration Specification for Cloud Applications) obeying the maximum, minimum, and increment of VNFs. Figure 4 depicts the averages for packet loss results of the four experiments considering 95% confidence interval. With high CPU processing, the VNF controller can handle most of the new requests, but others are retransmitted resulting in packet loss. As expected, the proposal can provide much fewer packet losses when compared to a single controller approach. It is worth noting that the average loss value is kept almost similar regardless of the increase in the number of simulated cars.

Figure 5 shows packet loss and jitter results for a file transfer application using two controllers.

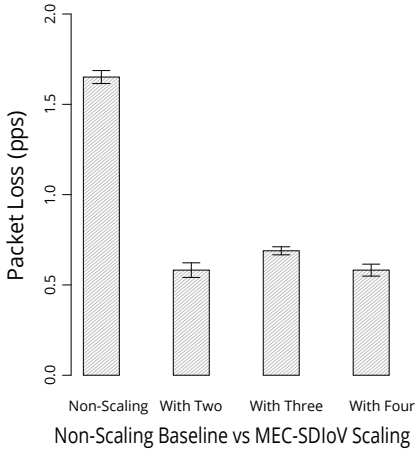

Fig. 4. Average Packet Loss per Second

Figure 5 (a) shows an average of 0.25 packets per second (pps) when the application was executed in the single controller environment and with 0.015 pps with the proposal. That is, it was possible to reduce packet loss by 16.6 times. Similarly, the Figure 5 (b) depicts a reduction on the jitter obtained with the adoption of the dynamic instantiation of a new controller. Compared to our scalable proposal, the gains are evident in terms of jitter control with respect to the non-scalable baseline. Figure 5 (b) presents the average values of 30 executions and corresponding 95% confidence intervals for both baseline and MEC-SDIoV solution. The concentration of data when the application was executed in both scenarios, first without scaling and then with the use of dynamic scaling of the SDN controllers. At the time the tests were performed without scaling, the mean and the interval represented in the graph are higher, showing that there is always a variance in the packet delay. According to the presented results, there is an impact on vehicular applications. That is, the vehicular network can be degraded due to congestion or high processing demands, which may impact in the overall QoS of vehicular applications. Thus, the solution proposed here can be used to benefit applications sensitive with stringent delay requirements since its devised to be a dynamic, flexible, and scalable SDVN running at the MEC.

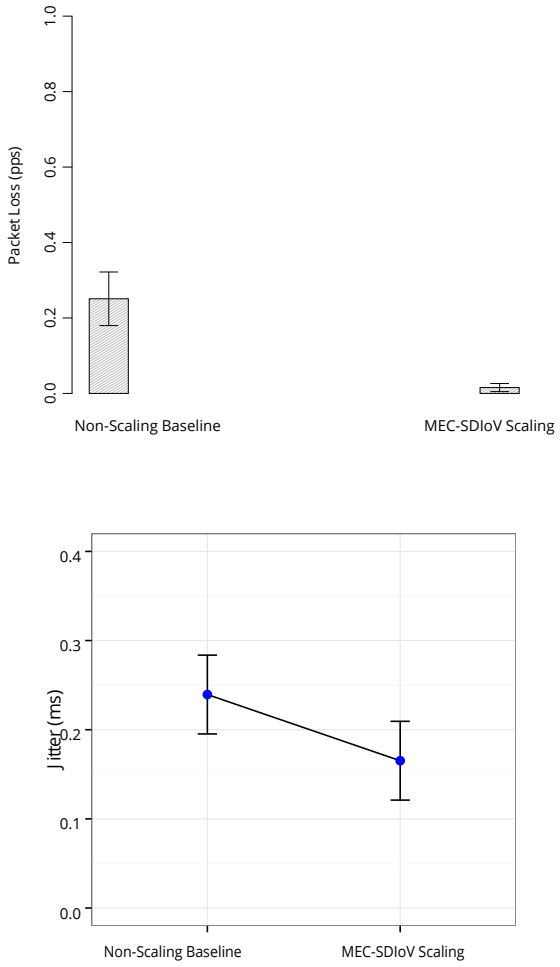

Fig. 5. Application Packet Losses and Jitter

V. Conclusion

This article presented a flexible architecture for the management of software-defined vehicular networks using NFV with the aim of instantiating SDN controllers dynamically. We have implemented a complete NFV architecture with its components according to the ETSI standards. A preliminary performance evaluation was carried out through an integrated testbed environment combining the real implementation of ETSI MANO in an Openstack platform along with IoV scenarios using Mininet-WiFi emulator and SUMO. The results showed that the proposal performs satisfactorily and allows the support of scalability by alleviating the congestion of the vehicular networks while granting adequate levels of QoS in terms of packet loss and jitter.

Future works consist on improving the monitoring of VNFs, since Openstack telemetry services adopted by our testbed collect a few cloud metrics related to the virtual machine. In addition, we intend to augment the evaluated scenario with more RSUs, mobility models, and video stream applications. Finally, with the limited cloud environment of our testbed in terms of processing and memory, it was possible only the increment of one VNF for each scaling out action. Thus, our future work will use a more powerful cloud testbed.

References

[1] Chahal, M., Harit, S., Mishra, K. K., Sangaiah, A. K., and Zheng, Z. (2017). A survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases. *Sustainable Cities and Society*, 35:830 – 840.

[2] Correia, S., Boukerche, A., and Meneguette, R. I. (2017). An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, 55(7):80–86.

[3] Fontes, R. D. R., Campolo, C., Rothenberg, C. E., and Molinaro, A. (2017). From theory to experimental evaluation: Resource management in software-defined vehicular networks. *IEEE Access*, 5:3069–3076.

[4] Kreutz, D., Ramos, F. M. V., Verssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

[5] Ku, I., Lu, Y., Gerla, M., Gomes, R. L., Ongaro, F., and Cerqueira, E. (2014). Towards software-defined vanet: Architecture and services. In *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 103–110.

[6] Liyanage, K. S. K., Ma, M., and Chong, P. H. J. (2018). Controller placement optimization in hierarchical distributed software defined vehicular networks. *Computer Networks*, 135:226 – 239.

[7] Palattella, M. R., Soua, R., Khelil, A., and Engel, T. (2019). Fog computing as the key for seamless connectivity handover in future vehicular networks. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, pages 1996–2000, New York, NY, USA. ACM.

[8] Sudheera, K. L. K., Ma, M., Ali, G. G. M. N., and Chong, P. H. J. (2016). Delay efficient software defined networking based architecture for vehicular networks. In *2016 IEEE International Conference on Communication Systems (ICCS)*, pages 1–6.

[9] Vilalta, R., Va, S., Mira, F., Casellas, R., Muoz, R., Alonso-Zarate, J., Kousaridas, A., and Dillinger, M. (2018). Control and management of a connected car using sdn/nfv, fog computing and yang data models. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 378–383.

[10] J. Wang, B. He, J. Wang, T. Li (2018). Intelligent VNFs Selection based on Traffic Identification in Vehicular Cloud Networks. In *IEEE Transactions on Vehicular Technology*

[11] Wan, J., Liu, J., Shao, Z., Vasilakos, A. V., Imran, M., and Zhou, K. (2016). Mobile crowd sensing for traffic prediction in internet of vehicles. In *Sensors*.

[12] Zhang, N., Zhang, S., Yang, P., Alhussein, O., Zhuang, W., and Shen, X. S. (2017). Software defined space-air-ground integrated vehicular networks: Challenges and solutions. In *IEEE Communications Magazine-2017*.

[13] Sakiz, F. and Sen, S. (2017). A survey of attacks and detection mechanisms on intelligent transportation systems: Vanets and iov. *Ad Hoc Networks*, 61:33 – 50.

[14] Giust, F., Sciancalepore, V., Sabella, D., C. Filippou, M., Mangiante, S., Featherstone, W., and Munaretto, D. (2018). Multi-access edge computing: The driver behind the wheel of 5g-connected cars. *IEEE Communications Standards Magazine*, 2.

[15] Naohisa, M., Kenki, T., Sho, H., and AOKI Hiroki, A. A. (2016). Iot network implemented with nfv.

[16] ETSI, N. (2013). Network functions virtualization - architectural framework.