An optimized application-context relocation approach for Connected and Automated Mobility (CAM)

Nina Slamnik-Kriještorac, Steven Latré, and Johann M. Marquez-Barja
University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Belgium
E-mail: {Nina.SlamnikKrijestorac, Steven.Latre, Johann.Marquez-Barja}@uantwerpen.be

***Abstract*—In this paper, we study and present a management and orchestration framework for vehicular communications, which enables service continuity for the vehicle via an optimized application-context relocation approach. To optimize the transfer of the application-context for Connected and Automated Mobility (CAM) services, our MEC orchestrator performs prediction of resource availability in the edge infrastructure based on the Long Short-Term Memory (LSTM) model, and it makes a final decision on relocation by calculating the outcome of a Multi-Criteria Decision-Making (MCDM) algorithm, taking into account the i) resource prediction, ii) latency and bandwidth on the communication links, and iii) geographical locations of the vehicle and edge hosts in the network infrastructure. Furthermore, we have built a proof-of-concept for the orchestration framework in a real-life distributed testbed environment, to showcase the efficiency in optimizing the edge host selection and application-context relocation towards achieving continuity of a service that informs vehicle about the driving conditions on the road.**

***Index Terms*—application-context relocation, vehicular communications, orchestration, 5G ecosystem, service continuity**

I. Introduction and Background

The 5G ecosystem illustrated in Fig. 1, consists of the 5G Core and New Radio (NR), including the managed and orchestrated distributed edge network infrastructure. In such ecosystem, a vehicle is capable to collect the contextual driving information, thereby connecting to the Connected and Automated Mobility (CAM) services, located at the edge in order to keep the communication latency to a minimum possible level. In particular, to be less dependent on driver's actions, and to ensure higher safety, the vehicle needs to receive instructions from the network infrastructure in less than 100ms [1], which requires service availability close to the vehicles, i.e., in the edge infrastructure such as Multi-Access Edge Computing (MEC) platforms, as well as transferring the application traffic via 5G Uu interface [2]. Thanks to the Software Defined Networking (SDN) and Network Function Virtualization (NFV), MEC platforms can offer distributed cloud-native service deployments at a closer proximity, enhancing the user experience and increasing network performance. However, due to the high mobility of users, such distributed service deployment requires an agile reconfiguration and constant monitoring in order to maintain the service continuity. Thus, in this paper, we present a management and orchestration framework that enables service continuity in a highly mobile environment, with the reference to the 3^{rd} Generation Partnership Project (3GPP) architecture for enabling edge applications [3], and ETSI NFV MANO framework [4]. The service continuity

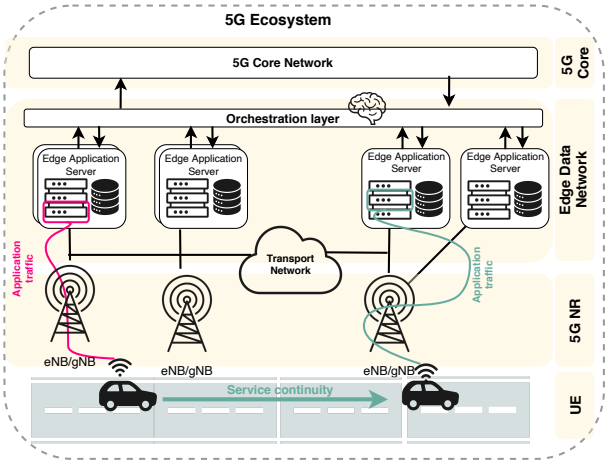


Fig. 1: Enabling CAM service continuity for vehicles in 5G ecosystem.

is enabled via an optimized application-context relocation approach that is triggered by a MEC application orchestrator while a vehicle, which is a consumer of the CAM service on the edge, moves along the road.

To efficiently solve the challenges on how and when to perform application-context relocation, the MEC orchestrator in our framework is performing the prediction of resource availability in edge NFV Infrastructure (NFVI), utilizing the prediction model based on Long Short-Term Memory (LSTM) [5], and making a decision on the optimal application service placement by running the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) algorithm, i.e., one of the widely adopted Multi-Criteria Decision Making (MCDM) concepts [6], thereby taking into account: i) the aforementioned resource availability prediction, ii) the latency and bandwidth on the communication path to the vehicle, and iii) geographical locations of vehicle and MEC host in the edge infrastructure. To measure the performance of the MEC application orchestrator, we have built a Poof-of-Concept (PoC) of the management and orchestration framework in a real-life distributed testbed environment, combining the Virtual Wall[1] testbed, and the Smart Highway[2] testbed, both located in Belgium.

[1]Virtual Wall: https://doc.ilabt.imec.be/ilabt/virtualwall/
[2]Smart Highway: https://www.fed4fire.eu/testbeds/smart-highway/

Fig. 2: 3GPP Architecture for Enabling Edge Applications.

Since the autonomous vehicles need to continuously collect the data from surrounding environment and network infrastructure, including the suggestions on braking and accelerating without driver assistance, the experimentation in our PoC re ects such a use case in which MEC application service is informing vehicle about driving conditions on the road (e.g., traf c jams, poor weather conditions, emergency situations, etc.). Thanks to the distributed service deployment, vehicle is being informed about driving conditions not only in its close proximity, but also in extended regions, thereby enabling vehicle to choose another route for its manoeuvre. The PoC is further described in Section III, where we also show the improvement in the application server response time when application-context relocation is performed, thereby proving the ef ciency of the MEC application orchestrator in optimizing the MEC host selection and application-context relocation towards achieving service continuity.

II. Application-context relocation

A. Management and Orchestration framework

As a part of Release 17, 3GPP is standardizing an architecture for enabling edge applications, while providing mutual awareness between edge client applications (i.e., in vehicle application), and edge application servers running in the edge data network. This 3GPP standardization track [3] created i) the application layer architecture, which is shown in Fig. 2, ii) procedures, and iii) information ows necessary for enabling edge applications over 3GPP networks. In particular, in architecture shown in Fig. 2, the edge network consists of i) Edge Con guration Server (ECS), which provides con guration data, i.e., Local Area Data Network (LADN) URI, to the Edge Enabler Client (EEC) to connect to the Edge Enabler Server (EES), ii) EES, which interacts with 3GPP core to collect network and service capabilities (e.g., location services, Quality of Service (QoS) management, etc.) that will improve the performance of edge application server, thereby enabling Edge Application Client (EAC) to connect to the server, and iii) Edge Application Server (EAS), which performs server functions and exchanges application data traf c with the client (Figures 1 and 2). On the client side, in our case in the vehicle, EEC discovers the edge network, retrieves the necessary information for connecting to the edge (e.g., coverage area/service area, types of application services or MEC applications, etc.), and connects to it via IP address provided by EES. Furthermore, different reference points, i.e., EDGE 1-EDGE 7, are de ned to enable communication between different architecture elements.

Fig. 3: Message Sequence Chart for the application-context relocation procedure.

In Fig. 3, we present the message sequence chart to showcase the operation of the application-context relocation from one edge to another, thereby mapping our management and orchestration framework (black boxes on the top), which is based on European Telecommunications Standards Institute (ETSI) NFV Management and Orchestration (MANO) [4] and presented in [7], to the 3GPP architecture for enabling edge applications (yellow boxes). In particular, when vehicle sends a discovery request to the MEC orchestrator, as a response, it receives a list of all available MEC application services that corresponds to the lters applied in the request. This way, the vehicle becomes edge-aware, as it can connect to any application server from the list. Once MEC orchestrator decides that vehicle needs to connect to another MEC application service due to e.g., increased resource consumption that will degrade the QoS, vehicle going out of the geographical service area, vehicle re-attaching from one User Plane Function (UPF) anchor to another, etc., the same reference point, i.e., EDGE-1, is used to inform vehicle about the newly selected MEC host (i.e., Relocation complete noti cation in Fig. 3). Furthermore, this noti cation contains the endpoint of the new MEC application instance running on the new MEC host, and client in the vehicle needs to be con gured in the way that it can dynamically change the IP endpoint of the application server from which it consumes the service.

B. Optimized MEC host selection

To transfer the context of application service that vehicle is consuming, and to enable this vehicle to continue utilizing the service in a seamless way, we need to i) identify a corresponding target MEC host, ii) perform transfer of application-context, iii) recon gure the traf c rules and management policies, and iv) setup a new communication path to the vehicle. The step i) is performed by our MEC application orchestrator that is designed as an extension of Kubernetes master role.

It runs the optimized MEC host selection algorithm, thereby predicting the resource availability in all MEC hosts that belong to the management and orchestration framework, by applying the LSTM based prediction. Furthermore, taking into account the predicted resource availability, the latency and bandwidth on the communication path to the vehicle, and geographical location of both vehicle and MEC hosts, the orchestrator makes decision whether application-context needs to be transferred to another edge or not, by performing the MCDM analysis. If the decision is made, and new node is selected for application placement, orchestrator instantiates new application service on the target MEC host, and allows application services from the source host to transfer the context to the target host, as shown in Fig. 3. Finally, once the context is transferred, the orchestrator sends a noti cation to the edge-aware client application in the vehicle, which then starts consuming service from the new MEC host, after the traf c rules and management policies are recon gured by the MEC application orchestrator. Due to the limited space in this manuscript, the detailed presentation of the prediction and MCDM algorithms is not included in this paper, but it will be part of an extended version.

III. Proof of Concept

A. Experimentation setup

The PoC that we have built to measure the performance of our MEC application orchestrator while performing application-context relocation (described in Fig. 3) is illustrated in Fig. 4a. The experimentation setup combines the components from two testbed facilities, the Virtual Wall testbed (Ghent, Belgium), which is a testbed for large networking and cloud experiments, and the Smart Highway testbed (Antwerp, Belgium), a test site built on top of the E313 highway for the purpose of Vehicle-to-Everything (V2X) research.

The detailed speci cation of testbed machines of MEC hosts 1, 2, and 3, is presented in Table I. The MEC hosts are utilized as distributed edge cloud environment where MEC application services are deployed, and the vehicle on-board unit is used as a client. The overall deployment is created in the Kubernetes[3] cluster, in which the Kubernetes master is deployed on a separate bare-metal server with the same characteristics as the other two Virtual Wall hosts that are used as worker nodes. Our MEC application orchestrator is running on the master node to extend the capabilities of Kubernetes master towards supporting optimized MEC host selection for application-context relocation. Thus, the NFVI in our PoC consists of three distributed MEC hosts, i.e., two bare-metal servers in Virtual Wall, and one GPCU inside the Road Side Unit (RSU) that is located on the highway site, all three running as worker nodes in the same Kubernetes cluster. Furthermore, we have enabled the Metrics API in Kubernetes cluster to collect CPU, memory, and storage consumption from all distributed worker nodes, in order to train and validate our prediction algorithm. The client in our PoC is deployed as a Docker-based web application, which is on-boarded within NUC in the on-board unit, and it is connected to the distributed MEC application services via long-range 4G.

[3]Kubernetes: https://kubernetes.io/docs/home/

TABLE I: System characteristics of the testbed machines.

PoC information				
Type	MEC host 1 RSU	MEC host 2	MEC host 3	Vehicle NUC
Testbed	Smart Highway	Virtual Wall	Virtual Wall	Smart Highway
Location	Antwerp	Ghent	Ghent	Antwerp
CPU (GHz)	1.280	2.252	2.252	1.9
RAM (GB)	32	48	48	8
Processor	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	2x 8core Intel E5-2650v2 @ 2.6GHz	2x 8core Intel E5-2650v2 @ 2.6GHz	I7-8650U
Storage (GB)	1024	250	250	8

TABLE II: The mean and standard deviation values for two scenarios.

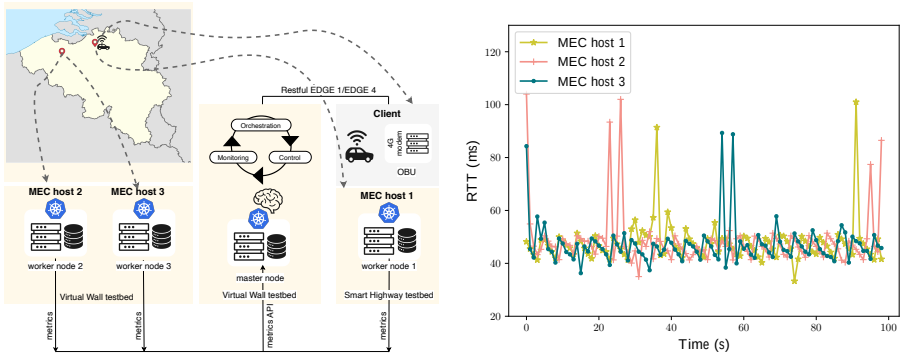
Scenario		Mean (ms)	Standard deviation (ms)
1	No application context relocation	331.117	117.543
2	Application context relocation	252.924	29.786

B. Results

The application service running on the distributed MEC hosts in our PoC are cloud-native Docker-based applications deployed in Kubernetes environment, with RESTful APIs exposed to vehicles for retrieving information about driving conditions on the road in a JSON format.

In Fig. 4b, we show the trace of the measured Round Trip Time (RTT) values for the client running in the vehicle on the Smart Highway, and for all three application servers deployed in distributed MEC environments, in order to test the impact of the network on the overall service response time, which contains the transmission and propagation delay (network impact), and computational delay on the application server (MEC impact). Furthermore, in Fig. 4c, we show the overall response time of the application server, measured on the client side, for two different scenarios. This response time is important because it shows the delay in retrieving the important contextual driving information from the server, and keeping this response time at a low level (e.g., below 100ms) is essential for vehicle to make decisions.

In both scenarios, the MEC host 1 is never selected by MEC application orchestrator for an application placement due to the high resource consumption (since we have increased it arti cially by performing load stress tests to train our prediction model), while MEC hosts 2 and 3 are being selected based on the projected resource consumption due to the RTT of similar scale. In the rst scenario no application-context relocation is performed, thus, vehicle remains connected to the MEC host 2, and as it can be seen in Fig. 4c, once the load increases on the MEC host 2 (after 200s), the response time of the application service is increasing, which means that the driving information about the conditions on the road might be signi cantly delayed at the vehicle side, leading to the inef cient decisions that will affect the whole manoeuvre experience. On the other hand, in scenario 2, we show that



(a) PoC setup. (b) RTT. (c) Response time.

Fig. 4: PoC in a distributed testbed environment.

in the case when load increases on the MEC host 2 (i.e., resource availability decreased), as predicted by our algorithm for the time after 200s, the proactive decision on relocating the application-context from application service on the MEC host 2, to MEC host 3, results in the relatively stable response time, which does not increase when vehicle starts retrieving service information from application service on the MEC host 2. Furthermore, a similar decision can be made by our algorithm in case user mobility event notification is received from the core network, and testing such scenario is part of our future work.

The mean and standard deviation values for both scenarios are shown in Table II, and we can see that in scenario 1, when there is no application context relocation for the observations that appear after 200th second, the deviation from the mean is large, i.e., the increase in response time is statistically significant. Thus, in scenario 2, we show that optimized and proactive MEC host selection that results in application-context relocation helps to improve the overall response time, and to prevent service unavailability that leads to outdated information about the conditions on the road, which consequently highly affects the manoeuvre decisions made by vehicle.

IV. Conclusion

In this paper we presented the management and orchestration framework for vehicular communications, based on the 3GPP architecture for enabling edge applications, and ETSI NFV. Such framework enables service continuity for the vehicles by performing an optimized application-context relocation from one edge host to another, thereby allowing vehicle to always connect to the most suitable application server to retrieve the important information about driving conditions on the road. This information is important especially for autonomous vehicles that need to derive decisions about maneuvering without any assistance from the driver. In the experiments on top of the PoC that we created, we show that optimized and proactive application-context relocation helps to improve the overall response time, and to prevent longer delays that cause the outdated information about the conditions on the road.

V. Acknowledgement

This work has been performed in the framework of the H2020 project 5G-CARMEN co-funded by the EU under grant agreement No. 825012. The work has been also supported by the Horizon 2020 Fed4FIRE+ project, Grant Agreement No. 723638. The views expressed are those of the authors and do not necessarily represent the project.

References

[1] J. Zhang and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2020, doi: https://doi.org/10.1109/JPROC.2019.2947490.

[2] A. Kanavos, D. Fragkos, and A. Kaloxylos, "V2X Communication over Cellular Networks: Capabilities and Challenges," *Telecom*, vol. 2, no. 1, pp. 1–26, 2021, doi: http://dx.doi.org/10.3390/telecom2010001. [Online]. Available: https://www.mdpi.com/2673-4001/2/1/1

[3] "3GPP Architecture for Enabling Edge Applications, howpublished = https://www.3gpp.org/ftp/specs/archive/23_series/23.558/, note = Accessed: 2021-04-12."

[4] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," *ETSI ISG NFV, ETSI GS NFV-MAN 001, V1.1.1*, 2014, online [Available]:https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.

[5] J. Violos, E. Psomakelis, D. Danopoulos, S. Tsanakas, and T. Varvarigou, *Using LSTM Neural Networks as Resource Utilization Predictors: The Case of Training Deep Learning Models on the Edge*, 12 2020, pp. 67–74, doi: http://dx.doi.org/10.1007/978-3-030-63058-4_6.

[6] A. Singh, "Major MCDM Techniques and their application-A Review," *IOSR Journal of Engineering*, vol. 4, pp. 15–25, 05 2014, doi: http://dx.doi.org/10.9790/3021-04521525.

[7] N. Slamnik-Kriještorac and J. M. Marquez-Barja, "Unraveling Edge-based in-vehicle infotainment using the Smart Highway testbed," in *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, pp. 1–4, doi: http://dx.doi.org/10.1109/CCNC49032.2021.9369622.