MEC Support for 5G-V2X Use Cases through Docker Containers

Claudia Campolo, Antonio Iera, Antonella Molinaro, Giuseppe Ruggeri
DIIES Department – University “Mediterranea” of Reggio Calabria – Italy
Email: {name.surname}@unirc.it

***Abstract*—The Multi-access Edge Computing (MEC) paradigm and the Cellular Vehicle-to-Everything (C-V2X) technology prove to be good candidates to address the vehicular applications' demands of high-performing connectivity and low-latency access to computing and storage resources. MEC provides cloud-like resources at the network edge, i.e., at the Multi-access Edge (ME) host. While vehicles move around, a service application instance running on a ME host may be triggered to move to another ME host to better support the application's demands. As a side effect, this migration may undermine service continuity.**

In this paper, we refer to the latest available ETSI MEC and 3GPP C-V2X specifications to investigate the issue of service migration between ME hosts in the context of vehicular communication. Early experimental results provide measures of the service migration latencies when ME applications run as Docker containers.

***Index Terms*—Multi-access Edge Computing; Vehicle-to-Everything; Docker; Container; 5G**

I. Introduction

Connected and Automated Vehicles (CAVs) represent one of the most significant transformations of the automobile industry that will influence and shape the future of road mobility. Emerging CAV applications, such as real-time situational awareness, high-definition local maps, cooperative sensing and maneuvering [1], [2] show hard-to-meet connectivity and latency demands.

Cellular Vehicle-to-Everything (C-V2X) is quickly gaining market traction among other candidate communication technologies for CAVs, under the leadership of major car manufacturers, telco companies, and chip makers [3]. The C-V2X rollout is facilitated by the ubiquity of the cellular infrastructure, its centralized orchestration and mature industrial foundation. Moreover, C-V2X has already established a clear and forward compatible path to fifth generation (5G) systems, holding promise of ensuring ultra-low latency and ultra-high reliability communication (URLLC), under high-density and high-mobility conditions.

CAVs applications entail process-intensive and low-latency computing and communication services. These characteristics make connection to the cloud often inadequate. Indeed, due to the long and fluctuating delay to reach the cloud, storing and processing data remotely prove to be unreliable and slow for many CAVs-related time-critical services, such as augmenting the visual perception of drivers to prevent collisions and accidents.

The Multi-Access Edge Computing (MEC) paradigm represents a low-latency alternative to the cloud, by pushing cloud-like (computing and storage) resources to the edge, close to where the data is being generated and also likely consumed, after being processed [2], [4], [5]. Thanks to MEC, the amount of (raw) data traffic traversing the core network is reduced, thus also relieving the pressure on the remote cloud infrastructure.

Compared to other MEC-assisted vertical markets, the automotive domain adds a layer of complexity to the process of computation offloading. It may entail the *migration of application services from one Multi-access Edge (ME) host to another*, in response to vehicles' mobility, so to ensure the shortest latency and the best edge service provisioning. Moving services closer to the users may cause, as a side effect, performance degradation and even interruption of ongoing services during the migration [6].

To reduce the impact of service disruption (*downtime*) during the migration, the European Telecommunications Standards Institute (ETSI) proposes *service pre-relocation* in an ongoing study on MEC support for V2X use cases [2]. The approach takes advantage of the knowledge of vehicles' trajectories in order to move applications towards the target ME host *in advance*, before the migration actually takes place.

In this paper, we elaborate on the service migration concept by also providing details about a migration strategy running on the open-source Docker[1] container platform. Docker supports fast development and running of distributed applications, independent of the underlying operating systems and with easy portability. Thanks to such features, Docker is considered as one of the most promising lightweight virtualization technologies and also a key enabler for MEC [7], [8]. The role of Docker in the automotive domain has been preliminarily investigated in [9], where Docker is used as a virtualization platform developed on board the vehicle.

In the context of Docker as an enabling virtualization technology for 5G-V2X ME applications deployment, this work takes a step forward and aims at providing the following main contributions:

- To specify a high-level architecture for the MEC support of 5G-V2X applications, which is aligned with the C-V2X specifications and the ETSI MEC reference framework.
- To design, among the functionalities of the architecture, a service migration procedure for V2X edge applications. It

[1]https://docker.com/

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.

exploits the modularity of the Docker platform with the aim to reduce the *service downtime*, i.e., the time during which the services offered by the virtualized migrating instance are not available. The main issues raising when dealing with service pre-relocation are neatly discussed.
- Going more practical and departing from the theoretical discussion in [2], to implement a preliminary analysis of the service migration performance for different Docker container sizes and network settings, which is intended to provide helpful guidelines for MEC-assisted 5G-V2X applications deployment.

The remainder of the paper is organized as follows. Section II provides background material for the conducted study. The proposed architecture for MEC support of 5G-V2X use cases is presented in Section III, whereas the designed service migration procedure is detailed in Section IV. Early results are reported in Section V, before concluding in Section VI with hints on future works.

II. Enabling technologies for 5G-V2X applications

5G entails crucial technology enablers for addressing the challenging requirements of several vertical markets, e.g., automotive, manufacturing, healthcare, city management, public safety. The most prominent 5G technologies for V2X applications are in the areas of enhanced connectivity in the Radio Access Network (RAN) segment and computing/storage services virtualization.

A. C-V2X: an overview

The C-V2X technology has been designed by the Third Generation Partnership Project (3GPP) to allow vehicles to communicate with everything, i.e., other vehicles (vehicle-to-vehicle, V2V), vulnerable road users (vehicle-to-pedestrian, V2P), roadside infrastructure (vehicle-to-infrastructure, V2I), and cloud/edge servers (vehicle-to-network, V2N) [10].

The technology uses the cellular network infrastructure to provide services to vehicular user equipments (VUEs) and enable enhanced direct V2V connectivity (be it network-assisted or unassisted) through the PC5 sidelink interface. The following entities have been added to extend the available architecture: the *V2X Control Function*, the *V2X Application Server* (V2X AS), and the *V2X Application*.

The V2X Control Function is the logical function for V2X network-related actions. It provides configuration parameters for VUEs located under the coverage of an eNodeB as well as out-of-coverage, i.e., in the absence of the cellular infrastructure.

The V2X AS has a wide range of functionalities including: the reception of uplink unicast data from the VUEs; the delivery of data to the VUEs in a target area using unicast and/or multicast interfaces. It may also provide centralized control and distribution of traffic, road and service information.

A V2X Application on board of a VUE allows it to communicate with the V2X AS. The IP address(es) of a (set of) V2X AS(s) in a given area can be either pre-configured in the VUE or announced by the eNodeB through downlink signaling to facilitate the interactions between the two counterparts.

B. The ETSI MEC architecture in a nutshell

In 2014, ETSI launched the Mobile Edge Computing Industry Specification Group (MEC ISG) to develop a standardized, open environment for efficient and agile integration of applications from different stakeholders across heterogeneous computing platforms at the edge of mobile networks.

Starting with use case driven demands, a reference architecture and a set of Application Programming Interfaces (APIs) for key MEC interfaces, along with the essential functionalities, have been specified in [11]. The main components of the MEC architecture are reported in Figure 1.

The *ME host* is an entity that contains the *ME platform* and a *virtualization infrastructure* which provides compute, storage, and network resources for the *ME applications*. The latter ones are running, e.g., as virtual machines (VMs) or containers, on top of the virtualization infrastructure provided by the ME host. They can interact with the ME platform to consume and offer ME services.

The *ME orchestrator* is the core functionality in the mobile edge system level management. It is responsible for maintaining an overall view of the ME system based on deployed ME hosts, available resources, available ME services, and topology. It is also in charge of selecting appropriate ME host(s) for application instantiation (and relocation) based on constraints such as latency, available resources and services, and also mobility.

The added value of MEC in V2X environments has been recently recognized by the 5G Automotive Alliance (5GAA)[2]. ME applications may leverage low-latency communication channels with the participating vehicles in the area, and virtually unlimited computing power compared to an in-vehicle embedded processor environment. They can also benefit from additional (context) information that is not directly available for the participating VUEs, e.g., via data fusion from multiple available sources [1]. Moreover, ETSI started to investigate V2X use cases in [2], specifically focusing on identifying the MEC features to support V2X applications. The document analyses relevant V2X use cases and identifies the new requirements, features and functions, with special attention on the support of vehicle mobility and its impact on service provisioning.

III. The reference architecture

Granted the aforementioned technology enablers, the reference scenario for our study is depicted in Figure 1.

A VUE (*VUE A*) runs a V2X Application (*V2X App A*) and may access both edge and cloud facilities through a V2N link towards the closest eNodeB. V2X edge applications run at the ME host and their lifecycle (i.e., instantiation, scaling, migration) is managed by the ME orchestrator. Among them,

[2]http://5gaa.org/

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.

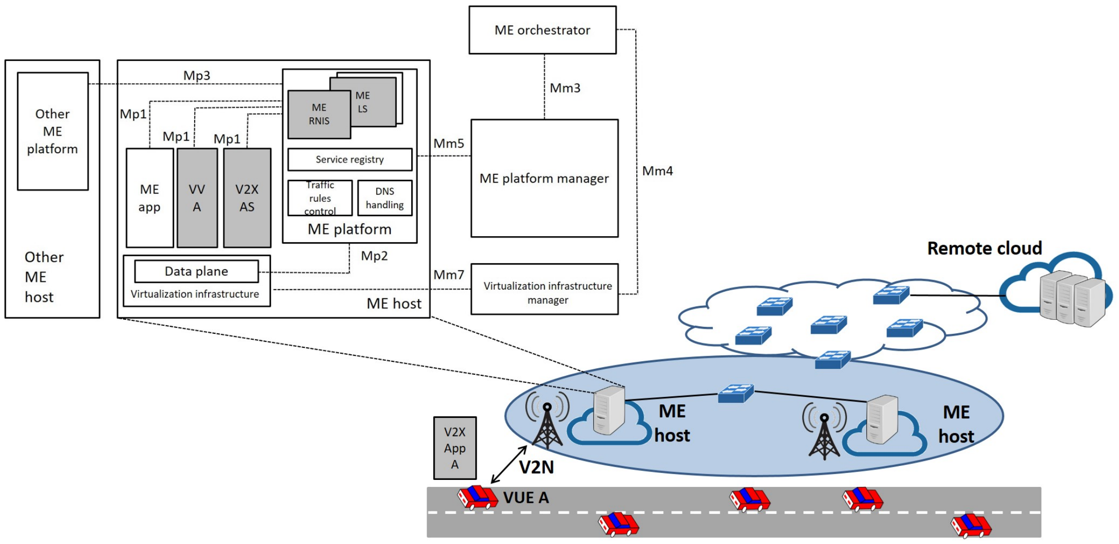

Fig. 1. Reference scenario. The functionalities and modules relevant to the targeted objectives are reported in grey boxes.

there is the 3GPP-defined *V2X AS*, deployed at the edge to reduce the access latency.

The reference architecture is aligned with C-V2X [10] and MEC framework [11]. In the following, modules in the envisioned architecture, along with their functionalities, are specified. The description is provided in the view of investigating the service migration procedures for 5G-V2X applications deployed at the edge, which is the focus of this work. For what concerns service migration procedures, we refer to the guidelines in ongoing ETSI MEC studies for the support of V2X use cases [2].

The V2X edge application. We assume that each V2X Application on board the vehicle has a corresponding counterpart in the MEC facilities. We refer to this application as the *Virtual Vehicle* (VV). In particular, we assume that VVs are hosted where also the V2X ASs are deployed.

The VV is intended to complement the V2X Application at the VUE with additional functionalities (e.g., analytics, data aggregation, video compression, object recognition) and with processing/storage capabilities. The VV builds upon the so-called *Virtual Object* concept, acting as a clone of constrained devices in Internet of Things (IoT) environments [12]. The VV is synchronized with the corresponding physical vehicle and works on behalf of it.

For the sake of portability, a VV can be deployed as a virtualized application, leveraging techniques such as VMs or containers. Here we assume that VVs are deployed as containers, which have the well-known advantages of being smaller than VMs and with a high starting speed [6]. Such a choice is also aligned with standardization efforts at ETSI that has created a work item in GR MEC 027 on *MEC Support for Containers* to identify the additional support that needs to be provided by MEC when ME applications run as containers.

The V2X Application on the VUE periodically feeds the corresponding VV, through the V2N link, with data retrieved by on board sensing devices, e.g., kinematics parameters provided by Global Positioning System (GPS) and accelerometer, videos/pictures of the surroundings captured by cameras. Floating Car Data (FCD) packets may be used to this purpose [13]. Such data may be either stored locally in the VV or processed before being consumed by the V2X Application (e.g., for object recognition purposes).

Stored data can be queried by other tenants besides the driver, e.g., transportation agencies, municipalities, car manufacturers, the 3GPP V2X AS. Tenants can query the VVs for data, with no need to resort to the VUEs. Some tenants could be interested in data aggregated over multiple VVs, such as in the case of a municipality that queries VVs about the road surface irregularities notified by vehicles in the region of their jurisdiction so to prioritize road repair [14]. Collected data could also help predicting congested urban areas [15] or assist the V2X AS for a more accurate and augmented situational awareness of the road users.

The ME orchestrator. It plays a crucial role in assisting the service migration. A VV running on a ME host may be triggered by the ME orchestrator to move to another ME host, to ensure quality of service demands, while the corresponding vehicle moves.

In particular, the ME orchestrator has the twofold task of: *(i) selecting the target ME host* (set of ME hosts) and *(ii)* deciding *when* and *whether* migrating the VV instance to it (them). To this aim, the ME orchestrator continuously monitors the status of ME hosts and the VUEs attached to them. For each VUE it tracks: *(i)* the current location, as exposed by the ME *Location Service* (LS), and *(ii)* the planned trajectory, as provided by the VV, that will be mapped in a sequence of traversed base stations. The transit time in each cell can be estimated by the knowledge of traffic road densities conditions and kinematics parameters, as well as through information about the radio network conditions provided by the ME *Radio*

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.

Network Information Service (RNIS) [11]. Both LS and RNIS ME services are native in the ETSI MEC platform [11].

Target ME host selection. Starting from the above information, the ME orchestrator identifies the set of candidate serving ME hosts on top of which the corresponding VVs can be instantiated and relocated. Indeed, at a given time, multiple ME hosts may serve a given VUE, especially if multiple (R)AN segments are available in the 5G system. The selection of the target ME host by the ME orchestrator entails a multi-criteria decision which should account for the following parameters: *(i)* the current load (e.g., CPU, storage) at each candidate ME host; *(ii)* the workload profile (e.g., CPU, storage demands) of the VV and its service requirements (e.g., access latency); *(iii)* the expected coverage time of the VUE under the candidate ME host, namely the RAN node providing access to it, and the quality of service guaranteed by the RAN node.

Migration timings and decision. For each target ME host (set of ME hosts), the ME orchestrator estimates the service migration timing. Such an estimation is particularly critical and depends on: *(i)* the workload profile of the VV (e.g., the VV size and the rate at which the VV is populated by the V2X Application on board the vehicle) and the service requirements (e.g., access latency, service continuity); *(ii)* the expected coverage time of the VUE under the candidate ME host; *(iii)* the considered VV virtualization deployment technique; *(iv)* the available throughput over the path interconnecting the serving and target ME hosts.

The first three parameters are either deterministically known or can be inferred with a good accuracy once the capabilities of ME hosts and the application profiles are known. The last parameter depends on the topology and the capacity of the links interconnecting the two ME hosts. However, it is reasonable to assume that it would not represent the bottleneck, since high-capacity backhaul links based on fiber and millimeter wave media can be considered. Moreover, the bandwidth can be guaranteed if a software-defined networking (SDN) approach is used over the path interconnecting the ME hosts.

It is worth to note that service migration is not always the best option. It could be the case when service migration is expected to be very frequent, hence incurring both high overhead and potentially long service downtime periods.

IV. The service migration procedure

Service migration may undermine service continuity. For this reason, solutions have been proposed in the recent literature with the aim of reducing the service downtime and ensuring a seamless service provisioning [16]–[18]. In [16], a proactive service replication is exploited to reduce the migration time of Docker containers. Service instances are replicated on multiple target ME hosts, which are kept synchronized before the migration takes place, hence resulting in a non-negligible overhead. Unlike that proposal, we specifically target the service migration *(i)* for V2X applications and *(ii)* to a single ME host. Selecting a single target ME host is viable for V2X applications, for which the planned or desired trajectory may be known [2]. This could be the case of a vehicle with pre-planned routes for a journey set by a car navigation system.

Near seamless service migration is also targeted in [17], where the Checkpoint/Restore In Userspace (CRIU) technique is successfully applied, but for Linux containers (LXC).

The solution proposed in this work is customized for the Docker environment. Indeed, we assume that the VV is deployed as a Docker container. Such a choice is because Docker allows easily building, managing, and removing a containerized application, with a small overhead and high flexibility. It also allows a dense virtualization of physical things at the edge [7], hence crucial for massively connected vehicles.

A. Main steps

The conceived service migration procedure follows the theoretical guidelines in [2] for what concerns the service pre-relocation concept and treasures the pre-copy technique in [18] at a practical level. To the best of our knowledge, at the time of this writing, there is no official migration tool for Docker containers. Thus, we build on the off-the-shelf Docker features to improve service migration procedures and reduce service disruption. Containers were originally designed to be stateless, i.e., they cannot maintain data when they are destroyed. A Data Volume (DV)[3] is a specially designated directory within one or more containers that enables data persistence, regardless of the containers lifecycle [19]. This layered Docker deployment facilitates the migration procedures.

The proposal includes two main steps, as detailed in the following and illustrated in Figure 2. For the reader's convenience, main notations are summarized in Table I.

1) **Service pre-relocation at the target ME host**. At time instant t_{pr} (i.e., T_{pr} seconds before the VUE is expected to leave the current ME host, at time t_h), the service pre-relocation is started. After being frozen, the VV container is migrated towards the target ME host. At this time, a new DV is created to store data provided by the V2X Application while the container is migrated.
2) **Service final relocation and access at the target ME host**. At time instant t_{fr} (i.e., T_{fr} seconds before the VUE is expected to leave the current ME host), the container is suspended and the remaining DV, populated in the VV container since the time instant t_{pr}, is transferred towards the target ME host. Then, the previously transferred VV container with the DV is booted at the target ME host. At time t_a, i.e., when the VV container is started on the target ME host, the VUE can connect to the target ME host. Thanks to pre-configured IP addresses of V2X ASs, seamless interactions can be enabled between the VUE and the services co-located with the V2X AS at the ME host.

B. Timing settings

The pacing of the envisioned steps is highly dependent on the application design and network/edge configurations.

[3]https://docs.docker.com/storage/volumes/

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.

TABLE I
TABLE OF NOTATIONS

Parameter	Description
t_{pr}	time instant at which the service pre-relocation, from the source to the target ME host, should start
t_{fr}	time instant at which the final service relocation should start
T_{pr}	time needed to complete the service pre-relocation, with the transfer of the container to the target ME host (service pre-relocation latency)
T_{fr}	time needed to conclude the service relocation, with the transfer of the last DV of the container to the target ME host
t_a	time instant at which the VUE can connect to the target ME host
t_h	time instant at which the VUE will handover to the target ME host
T_b	time needed to start the container at the target ME host
T_d	time during which the container cannot be accessed (service downtime)

T_{pr} should be computed in order to minimize the service downtime, T_d, which is equal to: T_{fr}+T_b. The sooner the service pre-relocation is started, the larger the T_{fr}. This is because, in such a case, a larger amount of data would feed the container at the serving ME host – after the frozen image has been already transferred towards the target ME host – which needs to be transferred when the container is stopped. In setting such parameters and, in particular, T_{pr}, it should be ensured that the service re-location procedure is not started too early; in other words, the target ME host should be predicted with a high accuracy. Indeed, variations in the workload of ME hosts, as well as varying road traffic conditions may affect the performed selection of the target ME host.

On the other hand, a too late triggering of the service pre-relocation, although accurate in terms of selected target ME host, may prevent the transfer of the VV container before the physical handover between the serving and target ME host occurs, hence affecting the service downtime.

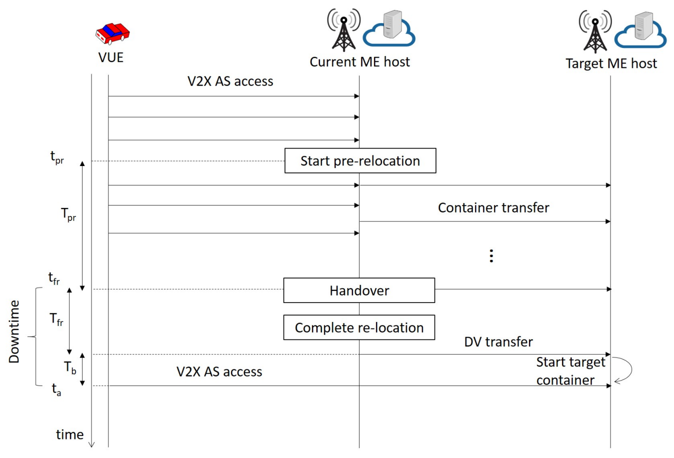

Fig. 2. Full workflow of service migration.

V. PROOF-OF-CONCEPT

A. Testbed description

A proof-of-concept (PoC) has been deployed to evaluate the practical realization of the proposed service migration strategy. In particular, a MEC scenario has been emulated with two ME hosts, hosting the VV container, respectively, initially and after the service migration. The ME hosts run over two workstations, which use the same operating system Ubuntu 16.04 LTS and are running a Docker Engine (version 1.21.2). They are both equipped with an Intel core i7 up to 3.40 GHz and 16 GB RAM. The two nodes are connected through a wired link and the *tc* utility [20] has been used to reproduce different bandwidth settings (i.e., C=100 Mbps and 1 Gpbs, to resemble congested and ideal network conditions, respectively). Results averaged over 10 experimental runs are reported.

The following metrics have been derived: *(i)* the *service pre-relocation time* (T_{pr}) and *(ii)* the *service downtime* (T_d).

The first metric has been derived as the time needed to transfer a container, by considering the following latency contributions: *(i)* the time needed to freeze the container (through the Docker *export* command); *(ii)* the time needed to transfer the container image (e.g., through the *scp* utility); *(iii)* the time needed to create a filesystem image for the container (through the Docker *import* command). The service downtime accounts for the following contributions: *(i)* the time needed to transfer the DV (e.g., through *dd* and *netcat* utilities[4]; *(ii)* the time (T_b) needed to run the complete container (the image of which is already stored at the target ME host) with the last transferred DV mounted (through the Docker *run* command).

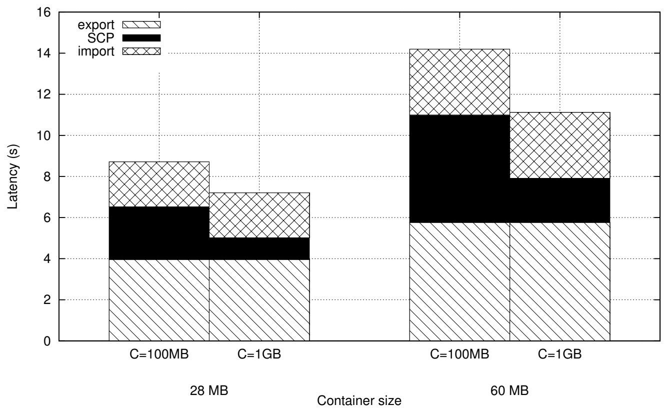

Fig. 3. Service pre-relocation latency: contributions for different container size and inter-edge link capacity (C) values.

B. Early Experimental Results

The service pre-relocation latency is reported in Figure 3. It reaches a value close to 15 s for a Ubuntu container image with size of 60 MB and lower than 10 s for a 28 MB-large Ubuntu

[4]The *rsync* utility can be used as an alternative, to further reduce the incurred latency.

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.

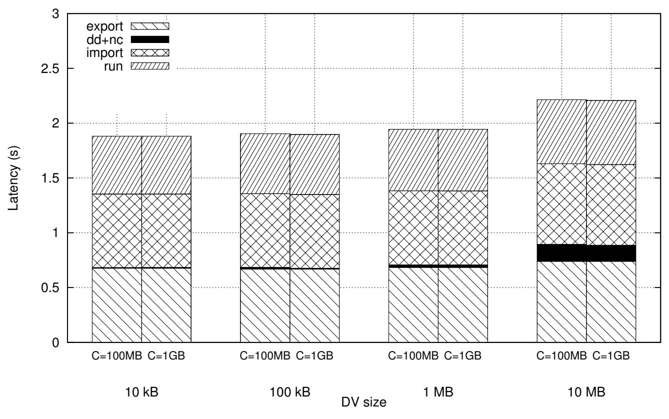

Fig. 4. Service downtime: contributions for different DV size and inter-edge link capacity (C) values.

container image. Latency values differ for the export, import and transfer operations, due to the different container size values. They provide a non-negligible contribution. However, it should be observed that the export/import operations highly rely on the capabilities of the hosting platform. The inter-edge link capacity also affects the overall latency contribution.

The service downtime is reported in Figure 4. Results show that the metric achieves values close to 2 s and they differ mainly for the DV transfer delay contribution. The contribution of the run operation is quite comparable in all cases. For a DV size of 10 kB, the downtime delay is below 2 s. This is a quite encouraging result. Indeed, the size of 10 kB is reasonable for a DV populated in around 10 s (the same order of magnitude of the measured service pre-relocation latencies) by 100 bytes-long FCD generated by vehicles and sent to feed the VV at a frequency of 1-10 Hz.

VI. Conclusion and Future Works

This paper preliminarily investigated the issue of service migration among edge nodes, with focus on 5G-V2X applications. A service migration procedure has been proposed which has been customized for a Docker-based ETSI-compliant MEC platform. The latter one hosts an edge-side vehicle clone accessed through V2N links. It complements V2X applications on board the vehicle and may serve multiple tenants accessing vehicular data. The proposal relies on the knowledge of the trajectories of vehicles to properly trigger service pre-relocation to a single target ME host and takes the best of the modularity of the off-the-shelf Docker containerization technology. Future works will target further reductions of the service downtime values, through more sophisticated copy and transfer techniques. Such improvements will allow to meet the urgent ultra-low latency needs of more challenging 5G-V2X applications, like those assisting (semi-)autonomous driving operations.

Under the circumstances that the trajectory of vehicles can be known in advance, hence, intuitively, facilitating the prediction of the target ME host and handover timings, other challenges may raise. The selection of the most convenient ME host, along with the prediction of the time instant on which the service pre-relocation should actually start, entail deeper investigations. The formulation of proper optimization models is in our research roadmap to target the trade-off between the service downtime and the accuracy in predicting the target ME host(s), e.g., in case of unpredictable changes of the trajectories, and pacing the service pre-relocation timings.

References

[1] "5G Automotive Association (5GAA) - Toward fully connected vehicles: Edge computing for advanced automotive communications," December 2017.

[2] "ETSI - Multi-access Edge Computing (MEC); Study on MEC Support for V2X Use Cases," September 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 Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, 2017.

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

[5] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Network*, vol. 32, no. 1, pp. 80–86, 2018.

[6] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23 511–23 528, 2018.

[7] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, 2018.

[8] G. Avino, M. Malinverno, F. Malandrino, C. Casetti, and C.-F. Chiasserini, "Characterizing Docker overhead in mobile edge computing scenarios," in *Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems*. ACM, 2017, pp. 30–35.

[9] R. Morabito, R. Petrolo, V. Loscri, N. Mitton, G. Ruggeri, and A. Molinaro, "Lightweight virtualization as enabling technology for future smart cars," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 1238–1245.

[10] "3GPP TS 23.285 v15.0.0. Architecture enhancements for V2X services. Release 14," March 2018.

[11] "ETSI GS MEC 003 v1.1.1. Mobile Edge Computing (MEC); Framework and Reference Architecture," March 2016.

[12] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the internet of things: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2016.

[13] O. Briante, C. Campolo, A. Iera, A. Molinaro, S. Y. Paratore, and G. Ruggeri, "Supporting augmented floating car data through smartphone-based crowd-sensing," *Vehicular Communications*, vol. 1, no. 4, pp. 181–196, 2014.

[14] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[15] F. Malandrino, C.-F. Chiasserini, and C. Casetti, "Virtualization-based evaluation of backhaul performance in vehicular applications," *Computer Networks*, vol. 134, pp. 93–104, 2018.

[16] I. Farris, T. Taleb, H. Flinck, and A. Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 4, p. e3169, 2018.

[17] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards a fast service migration in 5G," in *Proceedings of the NetSoft Conference*. IEEE, 2018.

[18] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2018.

[19] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[20] M. A. Brown, "Traffic control howto. [online]. Available: http://www.tldp.org/howto/traffic-control-howto/," 2017.

Authorized licensed use limited to: Dell. Downloaded on April 27,2022 at 15:13:41 UTC from IEEE Xplore. Restrictions apply.