Fast Service Migration in 5G Trends and Scenarios

Rami Akrem Addad[1], Diego Leonel Cadette Dutra[2], Miloud Bagaa[1], Tarik Taleb[1,4,5] and Hannu Flinck[3]
[1] Aalto University, Espoo, Finland
[2] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[3] Nokia Bell Labs, Espoo, Finland
[4] Centre for Wireless Communications (CWC), University of Oulu, Oulu, Finland
[5]Computer and Information Security Department, Sejong University, Seoul, South Korea

***Abstract*—The need for faster and higher-capacity networks that can sustain modern, high-demanding applications has driven the development of 5G technology. Moreover, low-latency communication (1ms - 10ms) is a key requirement of 5G systems. Multi-access Edge Computing (MEC) can be leveraged to attain the 5G objectives, since it allows the shift of part of services towards the vicinity of users, allowing the infrastructure to host various services closer to its end-users. Motivated by the evolution of real-time applications, we propose and evaluate two different mechanisms to improve the end-user experience by leveraging container-based live migration technologies. The first solution is aware of the users' mobility patterns, while the other is oblivious to the users' paths. Our results display a closer to 50% reduction on downtime, which shows the efficiency of the proposed solutions compared to prior works based on either a similar underlying technology, i.e., LXC or Docker.**

***Index Terms*—5G, Migration, Containers, NFV, MEC, SDN and Network Softwarisation.**

I. Introduction

The fifth generation of mobile communications will provide more than just a high data rate based on IP technologies, allowing the connection of billions of devices with fine-grained requirements [1]. Moreover, specific vertical industry services, e.g., automotive systems, e-health, public safety, and smart grids, have different service level agreements (SLAs) that must be taken into consideration to reach the envisaged 5G systems. Such an ambitious purpose demands an overall re-architecture of current 4G mobile networks. The new proposed architecture will allow the deployment of multiple logical networks over a shared infrastructure through the decoupling of the logical network from the physical infrastructure [2].

The adoption of Network Function Virtualization (NFV) [3] and Software-Defined Networking (SDN) [4] will help to concretize all elements mentioned before, as they represent the enabling technologies that have completely transformed modern network infrastructures. With SDN, network softwarization is able to provide a programmable network while using the NFV paradigm allows running Virtualized Network Functions (VNF) as software components on top of a virtualization system (e.g., Virtual Machines - VMs - or Containers) hosted in various clouds; allowing high flexibility and elasticity to deploy network services and functions. These VNFs will run on top of cloud computing environments distributed over the globe, which offers cost-effective services, scalability features, and multi-tenancy support, while possibly reducing both capital expenditures (CAPEX) and operational expenditure (OPEX) of 5G systems.

However, this architecture can be harmful to 5G systems' high data rates and low latency requirements if the NFV infrastructure is purely centralized, i.e., instantiating VNFs at faraway clouds introduces bandwidth limits and a higher end-to-end delay. The concept of Multi-access Edge Computing (MEC) [5] can overcome the limitation of centralized NFV deployments as it allows instantiating various VNFs, e.g., on top of containers, in the vicinity of users. The closer VNFs to the end-users are the higher data rates and lower latency we get. Furthermore, it is unusual for MEC nodes to dispose of enough computational resources for hosting standard virtualization technologies typically used in large data-centers, i.e., VM monitors or hypervisors. Therefore, due to the advantages in terms of management facilities, quick deployment and startup time [6], container technologies define an alternative technology in the MEC environment. Moreover, these technologies also allow faster replication [7], live service migration, and scaling methods than traditional VMs [8].

Even when considering the availability of several mechanisms and technologies such as the MEC architecture and the containers technology, users nowadays are everything except motionless, which induces a serious lack of flexibility and may take users far away from the original MEC node where their service started running. To overcome this problem, a new concept, dubbed Follow Me Cloud (FMC) [9], has been introduced. The FMC concept allows the mobility of services between different edges for placing them closest to end-users, which ensures low latency ($1ms - 10ms$) and high capacity (more than 100 Mbps). The stateful migration technique is used to enable the FMC concept by ensuring service continuity in case of the mobility of a service to a new mobile edge [10].

Meanwhile, autonomous vehicles and unmanned aerial vehicles (UAVs) are expected to be pillars for next-generation services and one of the 5G verticals. These emerging paradigms will leverage on MEC enabled infrastructure and container technologies, as they require a configurable network with high data rate and end-to-end latency below a user-specified threshold. By themselves, neither 5G systems or MEC ar-

chitecture mechanisms can ensure the required QoE for the end-users, been because of the UE's unpredictable paths or its mobility rate. Besides, the challenges facing autonomous vehicles, trains or UAVs can be a real threat to passengers' safety, demanding low latency connectivity.

Towards addressing the problem of service interruption related to known/unknown paths when migrating services between edge clouds, and based on the above-mentioned observations, the contributions of this paper are:

- The introduction of two migration solutions, the first solution assumes that the trajectories of mobile users are known, while the second solution is oblivious to the users' trajectory. We use the FMC concept in both solutions to ensure high availability and ultra-low latency;
- The design and the introduction of three different strategies for optimizing the disk migration related to the first solution related to the predefined path;
- The consolidation and the evaluation of these two solutions by leveraging the container technology to reduce the migration time as well as minimizing the services' interruption (downtime).

The remaining of this paper is organized as follows, Section II presents the motivation and background of this research. In Section III, we describe the types of migration evaluated in this paper and how they are deployed in our test environment. In Section IV, we present and discuss the results of our experimental evaluation. Finally, the paper concludes in Section V.

II. Motivation & Background

Live migration is the process through which we can transfer a running virtual instance between computer hosts, both its disk and current memory pages. Furthermore, while we may execute the disk copy phase with a running instance, the second phase, memory copy, must stop the virtualized instance, which creates a period where it will be unresponsive, also know as downtime. As, the number of memory pages, as well as the available bandwidth, are directly correlated with the instance's downtime duration, we can reduce the downtime using the concept of migration iteration. We can divide the copy of memory pages into several steps, each one of them, except the last one, can be done without stopping the virtualization instance and takes only the changes relative to the previous iteration. In our proposals, we named each one of these intermediate steps pre-dump phases, while the last one is named dump-phase stopping the instance. Our proposed solution leverages on the container technology dubbed Linux Containers (LXC) and CRIU (Checkpoint/Restore In Userspace) tool [11], instead of the legacy VM technology, as containers ensure a Linux system without the additional VM overhead. The checkpointing procedure consists in collecting and saving the state of all processes in the container. This action occurs during the last iteration, i.e., dump-phase, of the iterative migration or during the memory pages copy for a basic live migration. The restore procedure re-spawns these processes from a dump file we generate during the checkpoint.

Other researchers have investigated live migration based on lightweight containers. Among them, we highlight the works of Yang [12], who presented a generic checkpoint/restore mechanism and evaluated its performance using Docker (container technology). Each of the checkpoint and restore phases took $2183\ ms$ and $1998\ ms$, respectively, when considering a $256MB$ container size. For this reason, in this paper, we used the LXC container technology instead of Docker. Machen et al. [13] presented a multi-layer framework for migrating active applications in MEC. The authors leveraged the follow-me edge concept to enable both lightweight live migration (LXC) and pseudo-heavy live migration (KVM case). The authors expect these approaches to be at the core of an upcoming new generation of networks. They evaluated the containerized part of their work using LXC with different applications, e.g., video streaming, face detection, and game server. They were able to considerably reduce the total migration times albeit with a $2s$ downtime on average for a blank container. The increase in the downtime was due to the non-use of the iterative approach in the live migration process. Moreover, the authors disregard the impact of the known/unknown path on their results.

Concerning the previously cited works, in this study, we introduce a complete framework that handles both the known and unknown paths while optimizing all steps of the migration process, mainly the disk copy and the memory synchronization for achieving the 1 ms latency vision for the upcoming 5G and beyond mobile systems.

III. Fast service migration patterns in MEC environments

In this section, we describe our architecture for the proposed lightweight container migration framework, then we present the two proposed solutions for the asynchronous migration procedure. However, in more complex environments, VNFs are distributed similarly to micro-services architecture introducing traffic steering complexity. Addad et al. [14] proposed a method to handle all VNFs chain migrations, i.e., under the name of Service Function Chaining migrations, that takes into consideration both the synchronization the VNF chain's instances and the network resources consumption.

A. Main architecture and problem formulation

Fig. 1 depicts a three-layer cloud-based architecture for 5G networks. It supports scalable, distributed deployment models that aim to meet the 5G requirements, in terms of low latency and high data rate, for new mobile broadband and IoT services. The proposed architecture complies with ETSI-NFV standards and takes into account the orchestration and the management of the Core and MEC layers. The Core layer consists of a centralized computational power that can include data centers with powerful computing capabilities from different vendors, e.g., Amazon, Microsoft, or private cloud solutions based on open-source projects such as OpenStack. Orchestrated by the top layer, the MEC layer features the Radio Access Network (RAN) with high spectral efficiency and bandwidth. In the

NFV model, the Core and MEC layers form a distributed NFV infrastructure (NFVI) and would be controlled by one or more Virtualized Infrastructure Managers (VIMs). In the envisioned architecture, containers host the components of VNFs managed by numerous VNF Managers (VNFMs) that ensure the life-cycle management of all VNF instances spreading over multiple administrative domains (horizontal for MEC-to-MEC and vertical for Centralized Cloud-to-MEC).

Moreover, this distributed computing model allows users in the "users layer" to be close to the compute capabilities according to their mobility. In our presented use-cases, the users may be onboard of high-speed vehicles or UAVs, where their paths can be either pre-determined/predictable [15] or at a random path. The path-aware case allows us to trigger the migration process earlier, before reaching the edge of the cell, while in the path-oblivious solution the actual migration has to be completed before a given deadline, e.g., reaching the edge of the cell.

The main focus is the implementation of the live migration itself in several aspects to ensure a seamless migration across edge clouds, without taking into account other use-case-specific aspects, such as the signal strength received by each vehicle, user equipment (UE) or UAV. Moreover, we noticed that the ETSI-NFV orchestrator (NFVO) architecture should be augmented to handle fine-grained live migration decisions. Based on if the request is latency-sensitive or known/unknown path, the newly integrated module in the NFVO decides the best target location for the migration. The NFVO ensures all securities and communications requirements between VNFM source, VNFM destination, VIM source, and VIM destination. Given the **"single domain"** limitation expressed by both VNFMs and VIMs, the NFVO coordinates either directly with the VIM to request migration action from a MEC source to a MEC destination or through both the usage of VNFM and VIM elements, as both VNFM and VIM are considered as the executive part of the proposed approach.

Fig. 2 presents a flowchart that details our live migration strategies. We divided our proposed solution into two main parts: the disk and the memory migration phases. Initially, the need for a migration action is detected by the management plane, which can be a part of the ETSI-NFV's NFVO architecture. The management component can verify if the service is serving an instance with an unknown path, referred to by number 0 in Fig. 2 or in the case of known paths a series of methods are used, wherein the color of the rhombus signifies an action in a given part of the proposed architecture:

- The memory migration check (number 1 in Fig. 2) is the initial test where the availability of the clone image (base image and the application) is verified in the target MEC host. If available, a cloning process is started, followed by a memory migration;
- The partial migration check (number 2 in the same figure) is the next in case of non-availability of the clone image. Verification is done in the target MEC to find the base image, e.g., in the case of Ubuntu: trusty or xenial. Once found, cloning of the base image starts followed by a copy of the application data from the source MEC to the target MEC and a memory migration;
- The full migration check (refereed to by number 3 in Fig. 2) represents the final step and the worst-case scenario as the entire file system (rootfs), the application and the memory need to be transferred because of their absence in the target MEC.

However, for end-user with unknown paths, we must rely on the latency test as our second solution requires the use of a shared storage system, which may negatively impact latency-sensible applications. Since the container destination host delay to the shared storage can be bigger than the application tolerance, the migration can harm the service's QoS/QoE. To mitigate this issue, we can migrate latency-sensitive services to other MEC, even if that increases the migration time but offer a lower end-to-end delay.

In what follows, a detailed explanation related to known/unknown paths will be discussed. Initially, the predefined path will be presented, followed by the unknown path scenario.

B. Statefull service migration based on a predefined path

Hereafter, we present our solution for UE with predefined paths, previously discussed the path knowledge allow us to anticipate the different source and target MECs for any migration along with the mobility path of the UE. Moreover, we can implement the migration between MECs without the use of shared storage. It is worth mentioning that all previous checking, i.e, memory migration, partial migration, and full migration, works for the predefined path solution, as this solution is an optimization of the disk migration procedure, which is required to be transferred in known path solution. In the following section, we introduce an iterative live migration solution based on the CRIU [11] tool.

As described before, if both memory migration and partial migration checks failed, the pre-defined path solution starts first copying the container's file system along with the user files from the current MEC host to the destination MEC node using the **rsync** utility without service disruption. However, in case of a successful partial migration check, the application copy is the unique mandatory transfer. Otherwise (i.e. memory migration), no data transfer is required. Second, the memory of the container is iteratively copied from the source MEC host to the destination MEC host. In this step, the CRIU utility will be used for iteratively dumping the container's memory - while it is running - into a **tmpfs**-mounted directory at the source MEC host. Each dump is then copied to the destination host via the network into the **tmpfs**-mounted directory at the destination MEC host. Finally, the container will be restored at the destination MEC host. In this manner, we devise a control operation which is based initially on (number 4 in Fig. 2):

- a well-defined page number to guarantee the best time (downtime) possible;

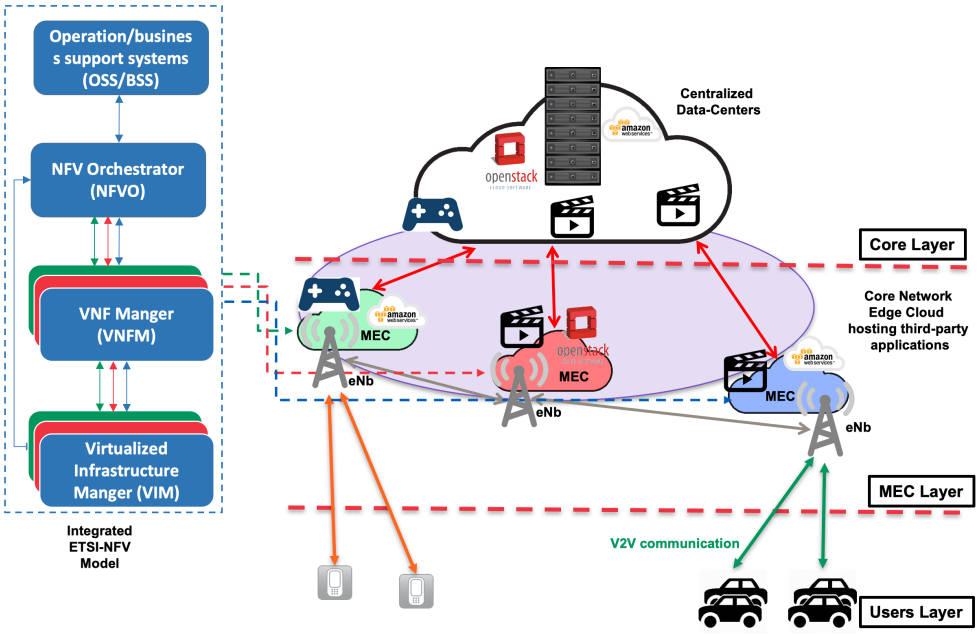


Fig. 1. Three layers cloud-based architecture.

- or use a fixed iteration number to avoid the scenario of the infinite loop where page numbers will never have a number below our threshold (the rate of modifying the dirty page is greater than the link speed).

C. Statefull service migration based on undefined path

In most real-world applications, the service provider (cloud service provider) does not know the movement patterns of the users. Accordingly, we propose a more generic solution that considers the paths of users to be unknown a priori. Under this condition, the copy of the file system and memory from the source MEC to the destination MEC could be a challenging process. We address this issue with a solution, named lightweight containers migration with a shared file system, that leverages an alternative, fast and efficient migration process. This pattern could be observed in Fig. 2 (number 0) when we consider an unknown path in addition to non-latency-sensitive services. First, we eliminate the need to copy files over the network during the migration phase, storing the container's file system along with the system images in a shared storage pool. This approach allows us to focus on the page memory migration process in the iteratively unload of the container's memory using CRIU on the source node and then immediately restoring the container to the target node. This approach uses more network resources while reducing the total migration time for LXC, thus the verification of latency-sensitive applications. We use the same logic as the pre-defined path is followed for handling the memory copy.

IV. Experimental Evaluation

We evaluate our envisioned container migration scenarios, using virtualized nodes, each node running Ubuntu 16.04 LTS with the 4.4.0-64-generic kernel, a 16 cores CPU, and 32GB of main memory. The interconnection among the nodes is set at 1Gb/s. We configure two testbeds container environment using LXC 2.8 and CRIU 3.11 to our evaluations:

- The first testbed consists of two VM hosts, each one representing a different Edge Cloud, i.e., an independent Infrastructure as a Service – IaaS – provider. Our container host is running on top of the first VM. We also deployed a third host representing the management plane previously discussed;
- The second testbed consists of three VMs. The first VM is the source MEC host, whereas the second one is the destination MEC host. Meanwhile, the third VM is the Network File Storage (NFS) server that we use to store the containers' file-system. Moreover, we ensure that in the testbed the three VMs can communicate among themselves to enable container migration. We use the communication between the MEC nodes and the NFS server for disk migration, while the direct communication between MEC nodes is used to migrate the memory content.

For every container migration, we evaluate the total migration time and the container downtime. The latter directly corresponds to the application of responsiveness/availability during the migration process. We conducted two sets of experiments, repeating them ten times each. The first one was a blank Linux container migration, with a file system size equal to 350 MB. We paid close attention to the container's network reachability throughout the migration process to observe the impact caused by adding persistent data. The second one was the migration of a video streaming server NGINX running on top of a container, whereby the file-system size was 590 MB. In both

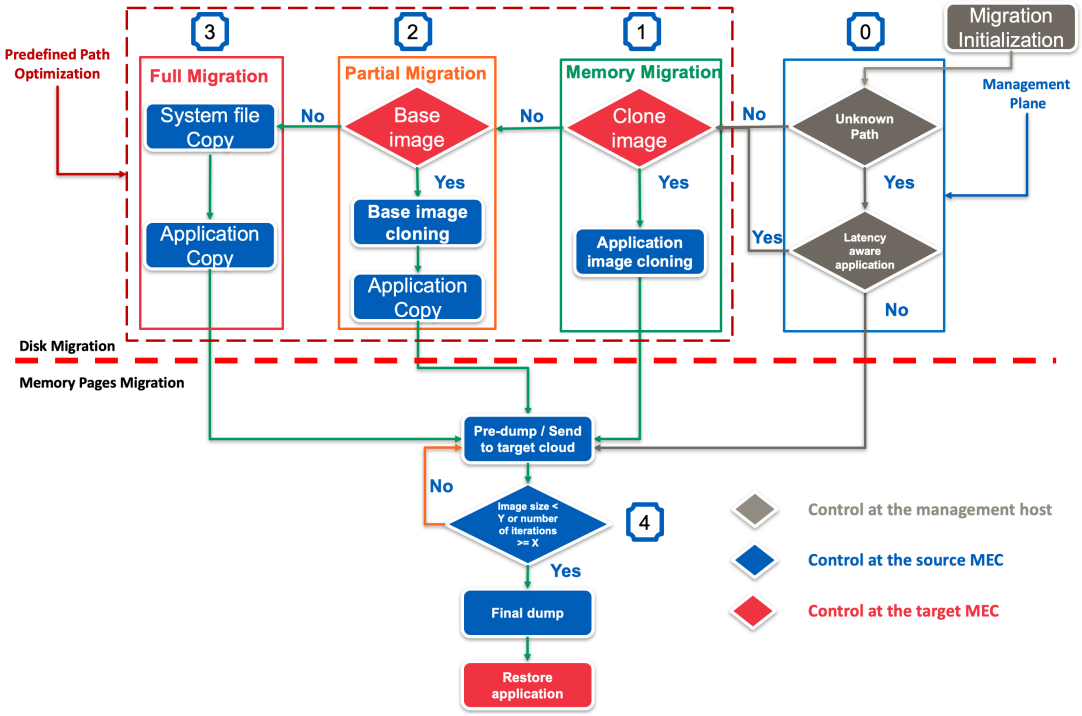


Fig. 2. Detailed live migration flowchart.

experiments, three strategies of migrations were evaluated: full migration, partial migration, memory migration for the predefined path case, and the shared migration as part of the unknown path situation.

A. Migration induced downtime

This experiment outputs the downtime, standard deviation, 95% confidence interval (CI), and coefficient of variation (CV) results for both blank and the video-streaming containers considering memory migration, partial migration, full migration as part of the predefined path solution and shared-file system migration as an unknown path solution. Detailed values are presented in Table I. As expected, the results for the video-streaming container are larger when compared to the blank container's results concerning all migration strategies. The difference in these results is due to the additional copies of the network connections status and the NGINX internal control data to the target cloud. We also noticed that the addition of the NGINX HTTP server introduced more variability in our experiments, nevertheless, this represented an increase in the CVs of less than 37.5%, 48.3%, 42.5 and 44.8% for memory migration, partial migration, full migration, and shared file system migration, respectively. From Table I, we can clearly observe that the downtime for the memory migration, partial migration and full migration related to the predefined path solution are quite similar which consolidate our proposal, i.e., those strategies are built to optimize the disk migration part of the predefined path, thus the migration of memory pages is not affected.

TABLE I
DOWNTIME COMPARISON IN CASE OF DIFFERENT MIGRATION APPROACHES.

Migration types	Strategies	Mean Time (s)	Std dev	CI 95%	Coef Var
Pre-defined path	Blank Memory-Mig	1.17	0.077	0.058	0.066
	Video Memory-Mig	1.238	0.218	0.165	0.176
	Blank Part-Mig	1.111	0.0645	0.049	0.058
	Video Part-Mig	1.382	0.166	0.126	0.120
	Blank Full-Mig	1.125	0.0640	0.048	0.057
	Video Full-Mig	1.327	0.177	0.134	0.134
Unknown path	Blank Shared f.sys Mig	1.825	0.123	0.092	0.067
	Video Shared f.sys Mig	2.454	0.073	0.055	0.030

In Fig. 3, we present a breakdown of the downtime to each migration procedure, the figure features the mean and the 95% CI of the times collected during the experiment for each approach, only the final iteration in addition to the restore phase was shown. Still in this figure, the memory migration, the partial migration, the full migration, and the shared migration can be viewed in the X-axis, while the Y-axis presents the time in seconds. Compared to the known path solutions, i.e., memory migration, partial migration, and full migration, the downtime for the shared file system migration increases. Our preliminary investigation suspects the network side because the target host restores procedure uses a remote file system which incurs an additional latency in both the final iteration and the restore procedures that form the downtime.

B. Total migration time evaluation

Our previous experimental results showed that our proposed approaches reduce the downtime caused by the migration

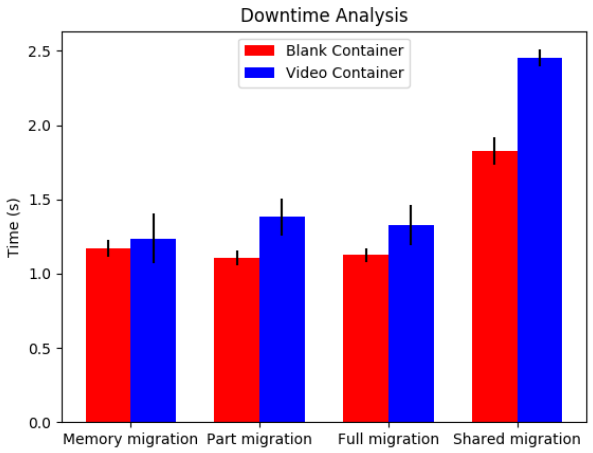


Fig. 3. Downtime comparison in case of different migration processes.

procedure. However, to enable its use for ultra-short latency services, we also need to address the total migration time. We addressed this evaluation using the same experimental scenarios of this section and plot the results in Fig. 4 for both the blank (red) and video-streaming (blue) containers. In Fig. 4, the Y-axis is in seconds. For each bar, we also plotted the 95% CI of the mean.

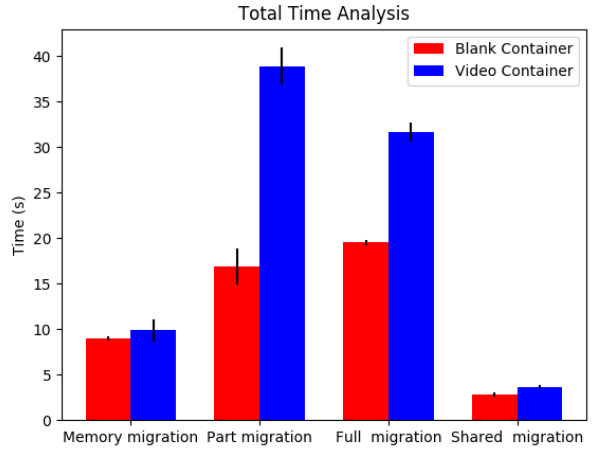


Fig. 4. Total migration time experienced in case of different approaches.

We present in Table II the mean total migration time, the Std deviation, the 95%CI, and the CV for the blank and video-streaming containers. For the pre-defined path solutions, the memory migration was the fastest as there is no need for disk copy, only a final "rsync" was leveraged to ensure the synchronization between the source and destination MECs, thus the additional $7s$ in the total migration time.

TABLE II
TOTAL MIGRATION TIME COMPARISON IN CASE OF DIFFERENT MIGRATION APPROACHES.

Migration types	Strategies	Mean Time (s)	Std dev	CI 95%	Coef Var
Pre-defined path	Blank Memory-Mig	8.990	0.271	0.204	0.03
	Video Memory-Mig	9.855	1.617	1.219	0.164
	Blank Part-Mig	16.885	2.605	1.964	0.154
	Video Part-Mig	38.918	2.67	2.013	0.069
	Blank Full-Mig	19.518	0.379	0.286	0.019
	Video Full-Mig	31.609	1.394	1.051	0.044
Unknown path	Blank Shared f.sys Mig	2.831	0.269	0.203	0.095
	Video Shared f.sys Mig	3.678	0.206	0.156	0.056

We focus on empty containers to show the impact of adding services have on migration time, from these results, we can observe that for all the pre-defined path migrations, the long migration time was due to the file system copy, which we avoided in the shared file system migration scenario. We must also highlight that for the video streaming, container size only increased the total migration time in the case with local storage, mainly due to the file system copy. Furthermore, for the shared file system scenario, the longer migration time of the video container, in comparison with the blank one, is due to the greater number of memory pages copied. Moreover, the partial migration strategy also induces a longer total migration time when we compare to the full migration procedure. Thus, we can conclude that sending the whole disk, i.e., rootfs or file system, through the network is faster than cloning the base image, e.g., trusty for Ubuntu distribution, followed by the transfer of the application's meta-data and data over the same network. However, we must highlight that partial migration is a more promising solution as it limits the bandwidth consumed during the migration process, as we only transfer the application's data.

C. Impact of the number of pages on the migration downtime

To avoid any bias from the underlying hardware technology on our experimental evaluation, during the last iteration we gathered the number of pages copied, and compute its mean, standard deviation (STD), and the 95% CI for both the memory migration procedure and the shared file system migration. We transferred for the blank container on average 500.7 and 517.8 pages for the memory and shared file system approaches, respectively. The standard deviation for both approaches were 2.83 and 17.56, with 95% CI of 2.31 and 14.30, respectively. For the video streaming container, the mean number of transferred pages was 706.6, and 551 with STDs of 61.07 and 16.8, respectively for the memory and shared file system approaches. Thus, we can conclude that the behavior exhibited for the "memory" and "shared file system" solutions are quite similar for both the blank container and our video streaming container, despite, the disparate downtime performance observed. The higher downtime for the shared file system solution is caused by the multiple small writes over the network. Meanwhile, the memory solution was able to drastically reduce the downtime as it keeps the number of copied pages in the last iteration small.

We also qualitatively evaluated the impact of the network bandwidth on experimental results where our best experi-

mental scenario was 500.7 $(2,050,867\ Bytes)$ and 706.6 $(2,894,234\ Bytes)$ pages transferred for the blank container and video container, respectively. As the amount of transferred data is unable to fully utilize a Gigabit link most of the downtime improvements are the consequences of our implementation and not the network itself since the network performance for small transfers is constrained by the TCP slow start and packet header/trail overhead. Moreover, even a simplistic analysis that disregards these issues, the data transfer would take around 164 ms for the blank container, and 231 ms for the video container assuming a Fast Ethernet connection (100Mbps), while for a Gigabit Ethernet (1Gbps) the time spent copying the pages over the network was of the order of 16.4 ms for the blank container and 23.1 ms for the video container.

V. Conclusion

In this paper, we proposed and evaluated four migration approaches based on the container technology to enable the Follow Me Edge concept using MEC, to ensure high availability and support ultra-low latency for real-time applications. While the first three approaches were used for the predefined path solution, the last approach was for the generic "unknown path" scenarios. The Follow Me Edge allows the system to guarantee a lower latency between the mobile user and the service provider, which is a fundamental requirement for Vehicular Networks (VN) and 5G networks. We have evaluated the proposed solution using real testbed experiments. Our results showed that while the shared file system approach delivered the shortest migration time, it also imposed the highest downtime. Meanwhile, we obtain a larger migration time in the approaches without a shared file system, because of the file system copy, which we did while the container was still running. We also showed that our iterative migration approach was able to achieve an average downtime of $1.111s$, an improvement of 61.039% in comparison to Yang [12], and 50% better than Machen et al. [13].

Finally, it is important to note that in the future, we plan to extend our evaluation to improve its performance by leveraging Artificial Intelligence techniques to decide the ideal time to carry out and enforce a migration operation.

ACKNOWLEDGMENT

This research work is partially supported by the European Union's Horizon 2020 research and innovation program under the MATILDA project with grant agreement No. 761898. It is also partially funded by the Academy of Finland Projects CSN and 6Genesis under grant agreement No. 311654 and No. 318927, respectively.

References

[1] NGMN Alliance, "5G White Paper," February 2015. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

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

[3] P. Mekikis, K. Ramantas, L. Sanabria-Russo, J. Serra, A. Antonopoulos, D. Pubill, E. Kartsakli, and C. Verikoukis, "NFV-enabled Experimental Platform for 5G Tactile Internet Support in Industrial Environments," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[4] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Software Defined Network Service Chaining for OTT Service Providers in 5G Networks," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 124–131, Nov 2017.

[5] 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 Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[6] Wubin, Li and Ali, Kanso, "Comparing Containers versus Virtual Machines for Achieving High Availability," in *2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 2015, pp. 353-358*.

[7] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[8] Y. C. Tay, K. Gaurav, and P. Karkun, "A performance comparison of containers and virtual machines in workload migration context," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2017, pp. 61–66.

[9] A. Aissioui, A. Ksentini, A. Gueroui, and T. Taleb, "On Enabling 5G Automotive Systems Using Follow Me edge-Cloud Concept," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2018.

[10] R. A. Addad, D. L. C. Dutra, T. Taleb, M. Bagaa, and H. Flinck, "MIRA!: An SDN-Based Framework for Cross-Domain Fast Migration of Ultra-Low Latency 5G Services," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, Dec 2018.

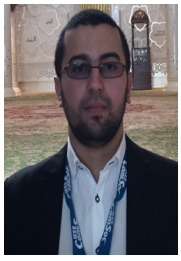
[11] T. CRIU, "Criu (checkpoint and restore in user space) main page," 2016. [Online]. Available: https://criu.org/Main_Page

[12] Yang Chen, "Checkpoint and Restore of Micro-service in Docker Containers," in *Proceedings of the 3rd International Conference on Mechatronics and Industrial Informatics*.

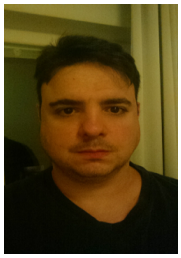
[13] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live Service Migration in Mobile Edge Clouds," *IEEE Wireless Communications*, vol. PP, no. 99, pp. 2–9, 2017.

[14] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards studying Service Function Chain Migration Patterns in 5G Networks and beyond," in *2019 IEEE Global Communications Conference, IEEE GLOBECOM*, Waikoloa, HI, USA, Dec 2019.

[15] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2577–2590, June 2015.



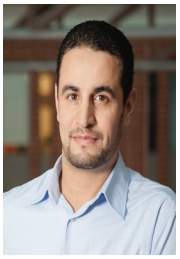
Rami Akrem Addad is currently pursuing the doctoral degree at Aalto University, Finland, in the Department of Communications and Networking, School of Electrical Engineering. He received the Licentiate degree and the Master degree from the University of Sciences and Technology HOUARI BOUMEDIENE, Algeria in 2015 and 2017 respectively, both degrees with high distinction and honors. He has been involved in European Project ANASTACIA Horizon 2020 for addressing cybersecurity concerns by leveraging SDN, NFV and Cloud architectures. His research interests include 5G network architecture, cloud-native technologies and approaches, network softwarization and slicing mechanisms, MEC, NFV, SDN, and distributed systems.



Diego L. C. Dutra is currently working as a professor at Federal University of Rio de Janeiro (UFRJ), Brazil, where he is also a member of the COMPASS Laboratory. He received a B.Sc. in Computer Science from UFF/Brazil, his M.Sc. and D.Sc. degrees in Systems Engineering and Computer Science Program from the Federal University of Rio de Janeiro, Brazil, in 2007 and 2015, respectively. He has worked as a postdoctoral researcher in the COMPASS/UFRJ and MOSA!C Lab/Aalto, from 2015 to 2016 and 2016 to 2017, respectively. His research interests include computer architecture, high-performance computing, virtualization, cloud computing, wireless networking, and Software-Defined Systems.



Miloud Baga received the bachelors, masters, and Ph.D. degrees from the University of Science and Technology Houari Boumediene Algiers, Algeria, in 2005, 2008, and 2014, respectively. He is currently a Senior Researcher with the Communications and Networking Department, Aalto University. His research interests include wireless sensor networks, the Internet of Things, 5G wireless communication, security, and networking modeling.



Tarik Taleb received the B.E. degree (with distinction) in information engineering in 2001, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is the founder and the Director of the MOSA!C Lab. He is the Guest Editor-in-Chief for the IEEE JSAC series on network Softwarization and enablers.



Hannu Flinck received the M.Sc. and Lic.Tech. degrees in computer science and communication systems from Aalto University (formerly, Helsinki University of Technology) in 1986 and 1993, respectively. He was with Nokia Research Center and the Technology and Innovation Unit of Nokia Networks in various positions. He is a Research Manager with Nokia Bell Labs, Espoo, Finland. He has been actively participating in a number of EU research projects. His current research interests include mobile edge computing, SDN, and content delivery in mobile networks, particularly in 5G networks.