MIRA!: An SDN-based Framework for Cross-Domain Fast Migration of Ultra-Low Latency 5G Services

Rami Akrem Addad[1], Diego Leonel Cadette Dutra[2], Tarik Taleb[1], Miloud Bagaa[1] 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

Abstract—**Given the constantly growing demand for inter-data-center services that 5G networks are bringing, live migration has become a covet and very challenging technology. Meanwhile, the emergence of Software Defined Networking (SDN) and Network Function Virtualization (NFV) technologies has completely transformed modern networks by offering more flexibility and at the same time more complexity. So far, investigations have been confined to integrating the live migration process with SDN/NFV paradigms in order to ensure the desired Quality of Experience (QoE). However, the simple integration is not sufficient to handle unexpected cases such as resources' unavailability, networking issues, and system control. For this purpose, we present MIRA!, a novel framework for managing reliable live migrations of virtual resources across different Infrastructure as a Service (IaaS), handling unexpected cases, while ensuring high QoS and a very low downtime without human intervention using an SDN aware solution. To validate our proposed framework, we performed a set of experimental evaluations under different configurations. The obtained results of our proposed framework show a 21% time reduction compared to a prior work and an interesting behavior while modifying the number of allocated CPU cores.**

I. Introduction

The emergence of Software Defined Networking (SDN) and Network Function Virtualization (NFV) technologies has completely transformed modern network infrastructures by enabling softwarization and efficient management of the network resources [1]. Softwarization represents one of the main enablers for the 5G's use cases [2], reducing the Capital Expenditures (CAPEX), the Operating Expenditure (OPEX), and allowing a flexible deployment schema. Softwarization achieves this through the use of virtualization, represented by the hardware and software virtualizations tools. The hardware virtualization tool refers to the conventional virtual machine (VM), while the software virtualization is lead by the lightweight container technologies. Meanwhile, the mass deployment of the connected objects, smart vehicles, and the Unmanned Aerial Vehicles have created the need for computing resources at the edge of the network, called Multi-Access Edge Computing (MEC) nodes [3]. However, it is unusual for MEC nodes to dispose of enough computational resources for hosting standard virtualization technologies typically used in large data-centers (VMs and servers). Therefore, due to the advantages in terms of management facilities, quick deployment and startup time [4], container technologies define an alternative technology in the MEC environment. Moreover, these technologies also permit a faster replication [5]–[7], live service migration and scaling methods than traditional VMs [8].

The aforementioned live migration procedure support is one of the essential features that allow us to use containers as a virtualization technique, as it is used across both physical machines and virtual machines to achieve load balancing, fault tolerance, and failure recovery. Live migration, in general, is comprised of at least two big steps: i) the copy of the disk; ii) and the copy of the memory pages. The two main approaches to live migration are Pre-copy and the Post-copy [9], and in both migration approaches, we have a period, called downtime, during which the service is unavailable.

In a conventional network, after each migration process, we consider a reactive approach for detecting the modification by using the concept of late detection of the new virtualization instances and the traffic is redirected using optimization algorithms such as Dijkstra or max flow principle. In another term, the switch will leverage MAC tables and the Address Resolution Protocol (ARP) protocol to find the new migrated instance; increasing the standard downtime in a critical situation such as live migration. However, an inter-data-center migration cannot be envisaged due to the impossibility of doing ARP broadcasting over the Internet [10]. A possible solution is to use the new trend of networking, namely SDN [11]. SDN provides a proactive approach to handle the flow resumption and eliminates the added discovery time. The flow resumption can be executed by the SDN controller in parallel with the last iteration in case of pre-copy migration (that we are using in our case) and before the restore part. This approach can enable the cross data-centers migration because of the ability to manipulate the switches even in a decentralized environment using the overlay network based on the VXLAN and the GRE technologies [12]. Nonetheless, a successful live migration demands more than the SDN technology.

In this context, a complete framework is needed to provide an autonomic migration and handle unexpected cases, e.g., similar internal IP address, the presence of a container with the same name on the destination host, and resource shortage in the target host which can destroy the entire system instead of regularizing it. In this paper, we present MIRA!, a novel framework to remedy the unexpected cases ensuring high QoS and delivering a lower downtime without human intervention. To the best of our knowledge, there is no existing framework able to handle all these cases.

The remainder of this paper is organized as follows. Section II outlines the related works. A design overview of the proposed framework and the implementation are presented in Section III. Section IV illustrates the experimental setup and discusses the obtained results. Finally, we conclude the paper and introduce future research challenges in Section V.

II. Related work

The migration based on the lightweight virtualization technology and assisted by an SDN controller constitutes one of the most suitable solutions to handle the ultra-low latency in the upcoming 5G technologies. Mann et al. [13] presented a network architecture that provides layer agnostic and seamless live and off-line VM mobility across multiple data centers. Their work leverages SDN and uses the principle of location independence to handle the inter-data-center limits. In some tests, they outperform the default Layer-2 approach by up to 30%. The proposed architecture was a valuable initiative to start working with the overlay network. However, as mentioned before, it was shown that software virtualization (system containers, application level container) achieves better overall time compared to the legacy VMs.

Machen et al. [14] presented a multi-layer framework for migrating active applications in MEC environment, using containers technologies instead of VMs. Their experimental results showed an average downtime of $2s$ in the blank container case. However, the authors were focusing on the system aspect of the migration without giving much interest to the network side of the migration, which means that they opted for a traditional network approach.

Bemby et al. [15] proposed ViNO (Virtual Network Overlay), an orchestration service used to create arbitrary network topologies with OVS (Open vSwitch) switches and VMs. ViNO creates an abstraction and allows the users to connect and use services through an overlay network, and enables as a use case a live migration based on container technologies and RYU SDN controller. Zhang et al. [16] extended the aforementioned work, describing both a theoretical model and a real testbed implementation. Their main focus was on the mathematical side of the problem. They showed the NP-completeness of the migration and proposed a heuristic algorithm to solve it. Nonetheless, in both works, we noticed the absence of a global framework that handles all the prerequisites in terms of system dependencies and the resource limits which usually influence the reliability of the migration process.

III. MIRA!: Design and Implementation

A. Design

Our proposal is a novel framework to manage live migration across multi-domain clouds and that is based on LXC containers [17]. The MIRA! framework provides functions to control container-based applications for many use cases including live streaming, enabling the Follow Me Edge Cloud (FMC) [18]–[20] concept and parallel migrations while ensuring the network scalability and guaranteeing the desired QoE. The proposed system allows the management of all basic operations in an SDN environment such as create, start, stop, and delete LXC containers through REST APIs or the available web interface. Furthermore, it enables the service migration over different edge clouds, by ensuring service connectivity through integration with SDN networking. Fig. 1 depicts the main components of the envisaged system.

1) Host Node (HO): The computational resources used to host a container in MIRA! can be a physical or a virtual node. Each host node must include the virtualization infrastructure, i.e., the LXC container engine and Open vSwitch (OVS). The service connectivity is guaranteed using OVS switches that are appropriately installed in the hosts and interconnected through VXLAN and GRE tunnels to enable the overlay network, decoupling the virtual network from the physical infrastructure.

2) The SDN controller: The SDN controller may run on top of a physical machine (i.e., dedicated server) or another VM that has more powerful resources in any cloud provider domain. In our implementation, we used ONOS [21], which can manage the OVS switches and installs the flows required for steering the traffic directed to the services.

3) MIRA! Global Orchestrator (MGO):

The MIRA! Global Orchestrator (MGO) is in charge of managing the container migration which is composed of several steps. First, a client can trigger the service migration through either a web interface or a high-level RESTful API by sending a request to MGO (step 1 in Fig. 1). After that, our framework enables a checking process, monitoring behavior, flows composition through the SDN controller and enables a parallel logic in order to allow a seamless migration as represented by steps 2, 3 and 4 in Fig. 1. Finally, we order the migration and the path redirection using the SDN controller (steps 5 & 6 in the architecture). All these features are provided by the MGO that is composed of different submodules:

a) Checking Module: Enables the verification of resources on both the source and target host before starting a seamless live migration of a service over multiple cloud domains. The collection of information on the host includes: i) the size of the disk; ii) the available memory, and finally; iii) the CPU. This analysis is necessary to check if there are enough resources in the destination node to host the migrated services while ensuring the desired performance.

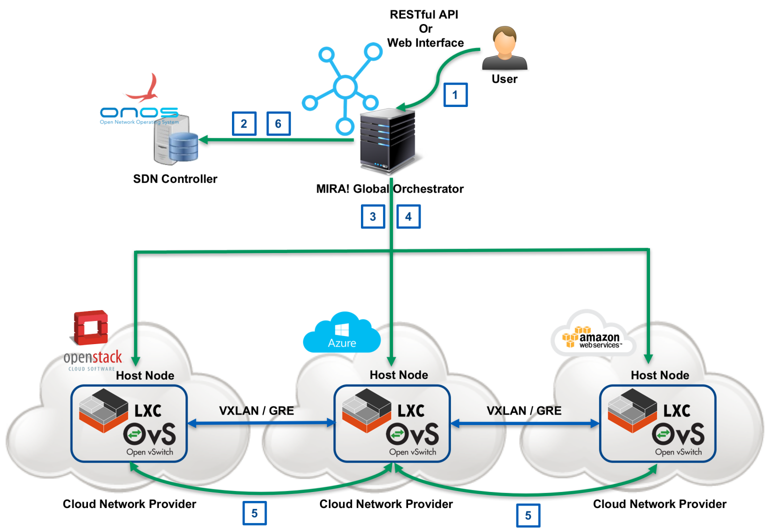

Fig. 1: MIRA! Architecture

Different strategies were developed according to the resource estimation:

- Conservative Approach: It considers the total utilization limits of the main memory and CPU on the destination node of all host containers, both interrupted and running. These values are then computed to verify that the host has enough resources to receive the services to be migrated. This approach ensures that containers are unable to interfere with the resources of others and guarantees the desired QoE to all users; i.e., even when a user starts his container off on the host, it will still have its resources reserved. However, this solution can involve an underutilization of the system, if the containers turned off are not reactivated.
- Semi-Conservative Approach : It only considers the limits of consumption of memory and CPU in the destination host from its running containers, and based on these values, it verifies if this host has enough resources to receive the to-be-migrated services. However, some issues in terms of resource shortfall can be experienced if stopped containers are reactivated, while different migration procedures are performed.
- Live Consumption Approach: It considers the current consumption of memory and CPU in the destination host of all running containers. These values are then subtracted from the free memory and the numbers of available CPU cores to verify that the host has enough resources to also host the to-be-migrated services. This approach aims to maximize the usage of resources in the destination host. However, potential resource shortage can be experienced, requiring appropriate countermeasures to balance the load in the system.

We developed these three approaches to guarantee both a trade-off between the execution time and the consistency of the resources in the MEC environment, and to also allow diversity of choice in the implementation phase.

b) Monitoring Module: Enables a live resource control on the source host and allows to trigger a seamless live migration of services across multiple cloud domains. Several thresholds, e.g., CPU, Memory, and Disk, can be fixed to enable an automatic migration. The autonomic migrations process can be used to attain the high availability needed for the different $5G$ use cases.

c) SDN Module: MGO integrates an SDN module to interact with the SDN controller component defined earlier. This module provides an SDN-based networking for both the normal communication and in case of handling the traffic redirection (path redirection) after the live migration. As discussed earlier, to remove the issue of the added delay due to the network legacy discovery mode (ARP request and MAC table), we leverage the Intent-based logic of ONOS to enable proactive path redirection [22].

d) Concurrent access & parallel logic Module: MGO guarantees concurrent access to the computational resources. The concurrency control must guarantee the correct execution of conflicting requests, e.g., a user does a migration of $container_X$, meanwhile, another tries to delete $container_X$. Furthermore, if enough computational resources are available, the MGO can enable the parallel live migrations of several containers.

e) Partial migration Module: The partial migration module was designed to check the availability of either base container images (for instance xenial, trusty and so on) or clone based images which provide a customized version of the image with installed applications (a typical example would be to have in addition to a base image of xenial, an application (NGINX web server) already present). Having such a module will be tantamount to drastically reduce the total time migration by either eliminating the need of base image transfer during the first part of the migration (copy of the disk) in case of the base image approach or eliminate the entire first part of migration and focus only on the second part represented by the memory copy. In the worst case scenario (neither the base image is present nor the cloned image is available), a full migration will be considered.

f) Retry Module for the efficiency of the live migration: The retry module is developed due to the need for high efficiency of the live migration. This module was designed to solve an issue that occurs in the last part of the live iterative migration in the second phase (memory copy). We observed failure cases in the last iteration. As a solution, we elaborated a mechanism able to detect the failure of the last step and trigger an automatic retry to ensure a highly efficient migration that meets the requirements of the 5G networks.

B. Implementation

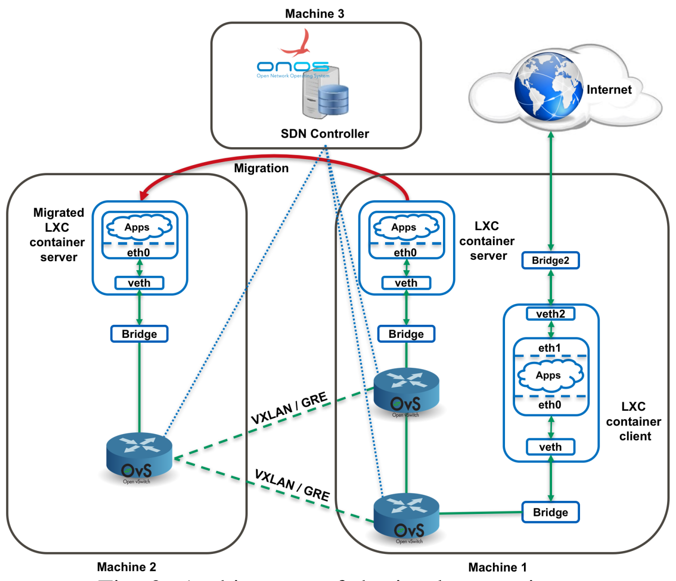

Fig. 2: Architecture of the implementation.

In the remainder of this section, we describe the implementation of the proposed framework. Our solution is based on an overlay network that uses SDN logic, which means that independently from the physical architecture, our solution takes advantage of the basic interconnectivity of data-centers and elaborates the required overlay network. In addition, our monitoring mechanism constantly checks the containers' state, and in case a predetermined threshold is reached, the system can activate an automatic migration across data-centers.

Fig. 2 shows the destination and the source virtual hosts (respectively Machine1 and Machine2). They contain all the virtualization software to deploy LXC container engine and programmable switches OVSs. The MIRA! framework is also running in Machine1 and is configured to implement the semi-conservative logic accounting for available resources in the implementation environment. Commonly, system-level containers (i.e LXC) work only with bridges, which are non-programmable level-2 switches (not suitable for SDN networking). To overcome this issue, we proposed a solution that works with multiple bridges per HO. In the proposed solution, we exploit the fact that each container has an inner network interface named (ethx) that may be mapped to a virtual interface (veth) in order to connect with the level 2 switch (i.e bridge) and each bridge related to a given container is connected to the OVS switches. The OVS switches are configured through the SDN controller, i.e., ONOS which is deployed in Machine3.

To link the SDN switches (OVS) to each other, taking into account the fact that they are situated in different physical hosts, we leverage on encapsulation protocols, i.e, VXLAN and/or GRE. We also added another interface (veth2 in the figure presented earlier) to the client container to allow Internet access.

1) MIRA!-based migration: The proposed container solution is based on an iterative migration procedure, which has been widely adopted for VM migration aiming to decrease the downtime during the migration process and is often known as pre-copy technique. In this framework, we use system commands provided by the CRIU utility and implemented verification steps to ensure the correct execution of pre-copy phase [23]. During each run, only the memory pages that have changed since the last run are transferred to the destination host. This technique can lead to great improvements in cases where the dirty page rate is lower than the transfer rate. Initially, we use a well-defined page number in order to guarantee the shortest possible downtime. Then, we use a fixed iteration number to avoid the scenario of the infinite loop, so to avoid the case when the system never reaches a number below the selected threshold, usually caused by the rate of page dirty being greater than transfer rate.

At the initialization of our algorithm, we established an initial end-to-end (E2E) path between a given client and the desired service. This step is carried out using the proactive approach enabled by SDN. When the migration procedure is issued, MIRA! starts the disk and memory copy using the CRIU tool. At the end of the last phase of the copy of the memory and before the end of the envisaged downtime, our framework interacts with the ONOS controller to ensure the continuity of the service by guaranteeing the management of the network side and establishing the E2E final connectivity before launching the restoration phase and cleaning (removal of the container in the source host). Our system adopts the Intent-based logic of ONOS in order to permit a proactive path redirection.

IV. Experimental evaluation

We experimentally evaluated our proposed framework using one physical server as shown in Fig. 3. The server has 48 cores with VT-X support enabled, 256 GB of memory, $1Gbps$ interconnection, and Ubuntu 16.04 LTS with the 4.4.0-77-generic kernel and QEMU-KVM installed. Each VM has a 16 cores CPU, 16GB of main memory, the same OS as the host node, and the container environment was based on LXC 2.8 and CRIU 2.6, both stable versions. Finally, we use an additional machine to access the test environment from the external network using virtual manager software. It shall be noted that our current experiments aim to show the behavior of the service migration under MIRA! supervision. Indeed, as we scale the number of nodes, this peer-to-peer migration holds. However, the shared network resources may increase both the downtime and the total migration time.

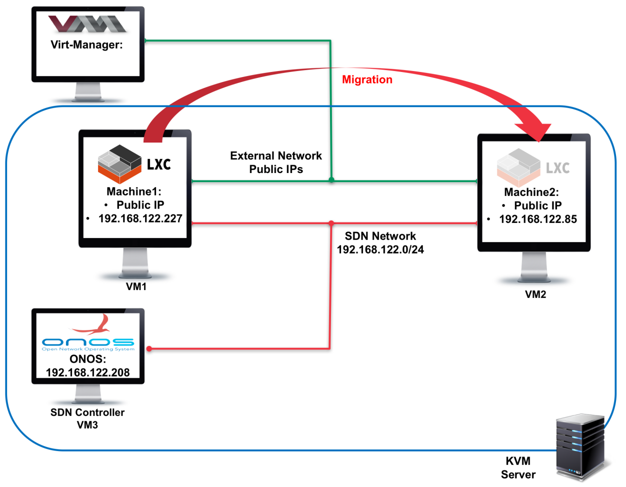

Fig. 3: Testbed Architecture.

In Fig. 3, the KVM server contains three virtualized computer nodes: machine 1 is the source from where the migration starts, machine 2 represents the target host and the ONOS machine represents the virtual machine hosting the SDN controller in order to handle the flow resumption.

The benchmark used in our evaluation is the streaming of a video with the characteristics described in Table I through an NGINX HTTP server. Moreover, we compared the performance of MIRA! against a Basic Solution that we developed based on an interactive migration using LXC and CRIU. Fig. 4 presents an overall comparison of the total migration time for both the Basic Solution and MIRA!. In this figure, we plot the OY-axis in seconds and the OX-axis shows each solution evaluated, the red bar represents the time it took to migrate the container between two hosts and the blue bar shows the control overhead imposed by MIRA!.

TABLE I: Test video specifications.

Type	Configuration
Codec	H.264
Duration	$442s$
Bit rate	$1,560\ Kbps$
Quality	$720p$
File size	$93.5\ MB$

The mean Total Migration Time of 10 executions for the basic solution was $28.7880s$, with Standard Deviation (STD) of $0.8729s$, a 95% Confidence Interval (CI) of $0.6582s$, and Coefficient of Variation (CV) 0.0303, Meanwhile, MIRA! mean Total Migration Time was $22.6271s$, that can be the breakdown on $20.3737s$ caused by the Migration process itself and $2.2544s$ of MIRA! Control Time overhead. The STDs for both Total Migration Time and Control Time is $2.1084s$ and $0.4562s$, with 95% CI of $1.5898s$ and $0.6334s$, respectively. It is important to note that with a CV of 0.2024, the Control Time shows greater variability during the experiments when compared to the 0.1034 of CV generated by Total Migration Time of MIRA!. The proposed framework was able to reduce the Total Migration time by about 21.4009%, although with a higher coefficient of variation, mainly caused by its control component which shows that our proposal is a promising solution. The achieved results in terms of Total Time migration reduction rely mainly on the developed partial migration module that was explained earlier in Section III-A3e.

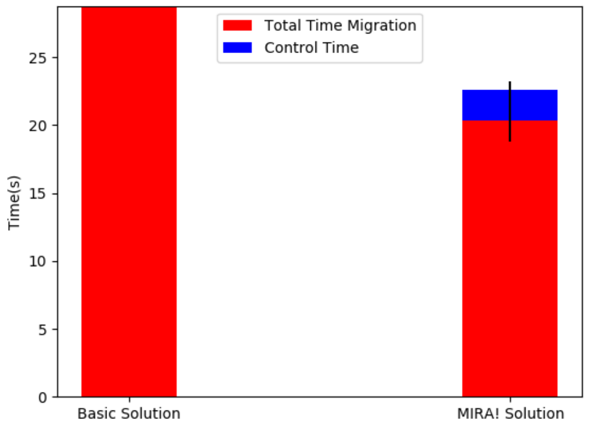

Fig. 4: Total Migration Time in case of the Basic Solution and MIRA!.

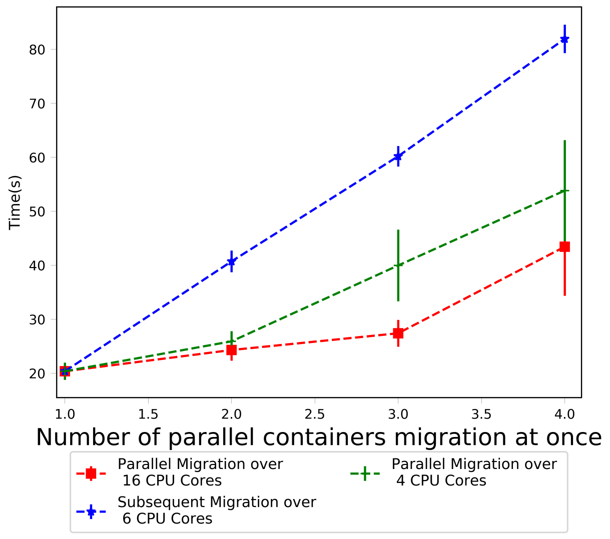

Fig. 5: Parallel migrations.

Fig. 5 shows how the number of parallel migrations impacts the total migration time when we vary the number of CPU cores. The OY-axis in seconds is the migration time, while the OX-axis is the number of parallel migrations. In Fig. 5, the blue curve shows the behavior of sequential migrations, the green curve presents the parallel migrations over two 4-core servers, and the red curve is the evaluation of MIRA!'s parallel migrations support over a 16-core hardware testbed. As anticipated, the sequential migrations show a linear behavior as we increase the number of migrations from one to four, which allows us to use this scenario as an upper bound on the total migration time. The 4-core testbed experiment manifested a distinctive linear behavior as to sequential migration since the total migration time had a slower increase rate for 2 migrations than for 3 and 4. Moreover, the results for the 16-core testbed exhibit a similar behavior seen in the 4-core case, albeit with the derivative changing after 3 parallel migrations. This behavior was caused as a result of the computational resource sharing between the containers and the MIRA! management system which limits the number of parallel migrations the server can support.

V. Conclusion and future work

In this article, we proposed a framework that enables live migration across data-centers belonging to multiple administrative domains. The framework ensures short downtime and supports service continuity leveraging SDN. In addition to the SDN paradigm, the proposed framework integrates a full mechanism to handle resources' unavailability, network issues, and system control. The MIRA! framework is fully developed in the MOSA!C Lab research group [24] and is available as an open source project on Github [25]. We also validated the proposed framework by implementing an SDN-based testbed. The obtained results serve as a basis for future research work, where we will consider more complicated scenarios based on arbitrary network topologies with a high number of OVSs (Open vSwitch) switches and VMs. In these scenarios, the SDN-based migration can be formulated as an optimization problem of multiple objectives and constraints with higher complexity. In addition, the mobility of end-users will be investigated to achieve the follow me edge cloud concept for the upcoming 5G mobile systems.

Acknowledgment

This work was supported in part by the Academy of Finland Project CSN under Grant No. 311654. The work was also supported in part by a direct funding from Nokia Bell Labs, Espoo, Finland.

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing; softwarization: A survey on principles, enabling technologies; solutions," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2018.

[2] N. Alliance, "5G white paper," Tech. Rep., February 2015. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

[3] 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.

[4] 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.*

[5] 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.

[6] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5g edge network," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[7] 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, e3169 ett.3169. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3169

[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] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *2nd Symposium on Networked Systems Design and Implementation (NSDI 2005), May 2-4, 2005, Boston, Massachusetts, USA, Proceedings.*, 2005. [Online]. Available: http://www.usenix.org/events/nsdi05/tech/clark.html

[10] U. Kalim, M. K. Gardner, E. J. Brown, and W. c. Feng, "Seamless migration of virtual machines across networks," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013, pp. 1–7.

[11] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.

[12] T. L. Foundation, "Connecting vms using tunnels," Tech. Rep., (Last visit on : 16-4-2018). [Online]. Available: http://docs.openvswitch.org/en/latest/howto/tunneling/

[13] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman, "Crossroads: Seamless vm mobility across data centers through software defined networking," in *2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 88–96.

[14] 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.

[15] S. Bemby, H. Lu, K. H. Zadeh, H. Bannazadeh, and A. Leon-Garcia, "Vino: Sdn overlay to allow seamless migration across heterogeneous infrastructure," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 782–785.

[16] S. Q. Zhang, P. Yasrebi, A. Tizghadam, H. Bannazadeh, and A. Leon-Garcia, "Fast network flow resumption for live virtual machine migration on sdn," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, Nov 2015, pp. 446–452.

[17] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sept 2014.

[18] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, September 2013.

[19] 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.

[20] A. Ksentini, T. Taleb, and F. Messaoudi, "A lisp-based implementation of follow me cloud," *IEEE Access*, vol. 2, pp. 1340–1347, 2014.

[21] O. team, "Onos," Tech. Rep., (Last visit on : 16-4-2018). [Online]. Available: https://onosproject.org/

[22] ——, "Onos," Tech. Rep., (Last visit on : 16-4-2018). [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework

[23] C. team, "Iterative migration," 2016. [Online]. Available: https://criu.org/Iterative_migration

[24] MOSA!C Lab research group , 2016. [Online]. Available: www.mosaic-lab.org

[25] ——, "MIRA! framework," 2018. [Online]. Available: https://github.com/MOSAIC-LAB-AALTO/MIRA-