Lightweight Service Replication for Ultra-Short Latency Applications in Mobile Edge Networks

Ivan Farris[1,2], Tarik Taleb[1], Antonio Iera[2], and Hannu Flinck[3]
[1] Aalto University, Finland
[2] University "Mediterranea" of Reggio Calabria, Italy
[3]Nokia Bell Labs, Finland
Emails: ivan.farris@aalto.fi, talebtarik@ieee.org, antonio.iera@unirc.it, and hannu.flinck@nokia-bell-labs.com

***Abstract*—Edge Cloud infrastructure will play a key role in extending the range of supported real-time cloud applications, by guaranteeing extremely fast response times. However, user mobility requires fast relocation of service instances, which represents an open challenge for resource-constrained cloudlets interconnected by high-latency and low-bandwidth links. In this paper, we investigate container-based virtualization techniques to support dynamic Mobile Edge Computing (MEC) environments. In particular, we design a framework to guarantee fast response time, by proactively exploiting service replication. A preliminary performance analysis is conducted to identify the possible advantages introduced by the proposed approach compared to classic migration procedures.**

I. INTRODUCTION

Cloud platforms have been receiving an ever-growing attention over the last years as a means to support applications belonging to a wide range of IT domains, such as multimedia, online gaming, and IoT. In particular, by offering on-demand processing and storage resources, the cloud environment provides the underlying infrastructure to execute flexible services and ease the deployment and updates of new IT applications. However, remote centralized cloud datacenters could suffer from limitations due to high latency and risk of workload and network bottleneck [1].

To face these issues, both academic and industrial research communities have focused on Mobile Edge Computing (MEC) solutions to exploit processing and storage capabilities at the edge of the network, as near as possible to the end-user [2]. In this regard, the deployment of micro data centers in the network access points, known as Cloudlets [1], can guarantee remarkable benefits in terms of low latency interaction and scalability, by balancing the workload over the distributed infrastructure. Furthermore, bringing the cloudlet concept into the IoT scenario results in the definition of the so-called Fog Computing paradigm [3] [4], where specific IoT-related challenges are considered. Not by chance, edge computing is considered one of the enabling technologies to guarantee the 1ms latency dream for the next-generation 5G networks.

On the other hand, MEC solutions introduce several challenges. First, accounting for the resource constraints of cloudlets, specific virtualization solutions and orchestration features should be used to deploy services in the distributed infrastructure. In this regard, container-based virtualization is considered one of the most promising solutions for MEC environments [5-8]. Containers implement isolation of processes at the operating system level of the underlying host machine, thus avoiding the overhead due to virtualized hardware requested by hypervisor-based virtualization. This enables fast initialization and dense deployment of services, as experimented in [9]. All these features make containers extremely attractive for usage not only in data-centers, but also in edge nodes, such as cloudlets and IoT gateways [10]. Nevertheless, the limited resource capabilities of these edge devices require exploring new forms of cooperation between neighbor nodes, to better meet peak service requests and to support load balancing [11].

Further challenging issues of MEC scenarios relate to user mobility. To guarantee service continuity and to meet the strict requirements of application latency, the MEC framework needs to properly cope with service migration between edge nodes. In this way, if a user moves out from the coverage/service area of a specific cloudlet, then the MEC system is able to promptly detect the user movement and relocate the user application by deploying a new service instance, hosted on a cloudlet closer to the end user. However, if this procedure involves time, especially when large data volumes must be migrated, then the user can experience degradation in the application performance.

In this paper, we focus on the investigation of novel approaches to better support user mobility in a federated MEC infrastructure. By exploiting the potential of emergent container-based virtualization techniques, we design methods based on lightweight service replication for stateless micro-services. The proposed approach aims to reduce the service migration time in comparison to traditional solutions and to increase service resiliency in case of host failure. The contributions of this paper can be summarized as follows: (*i*) we analyze and identify the challenges to support mobile service provisioning in a MEC environment; (*ii*) design a framework to support efficient service relocation by exploiting instance replication; and (*iii*) present a preliminary analysis which evaluates the possible benefits in comparison to classic reactive migration approaches.

The paper is organized as follows. Section II provides an overview of related research works. Section III illustrates the reference scenario and the envisaged solutions for service provisioning in MEC environments. Section IV introduces the design of the proposed framework, by defining the core modules and relevant functionalities. The performance evaluation is reported in Section V. Manifold open challenges are discussed in Section VI. Finally, conclusions are drawn in Section VII.

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

II. Related Work

A. Mobile edge computing and stateless applications

Edge micro-cloud infrastructure is an emerging paradigm to support computation/storage near to the end-users. This approach becomes unavoidable for real-time cloud applications with extremely strict latency requirements, as it allows for overcoming poor connectivity and long delay due to remote datacenters. Two aspects play a key role in the deployment of effective solutions for MEC scenarios: (i) virtualization technologies and (ii) service continuity due to user mobility. For the former point, hypervisor-based virtualization techniques present remarkable overhead in terms of both processing and storage capabilities. Furthermore, they involve high latency for start-up activation and migration procedures. Container-based virtualization enables high density deployment of services, but presents limited features to support stateful service migration between different host nodes.

Nevertheless, MEC-oriented application models have given a considerable impetus towards the deployment of stateless services. Whereas for stateful applications, the state of the application and the network stack must be preserved in case of migration or failure, stateless applications do not foresee internal stored sessions and, instead, they rely on user inputs or distributed shared storage. The stateless feature introduces several advantages in terms of flexibility, scalability, and reliability. Indeed, a stateless application could be replicated on different worker nodes and, based on the specific offloading request and the current connectivity quality, the most appropriate instance could be selected. The concept of stateless application is considered one of the pillars of the micro-service architecture style [12], a novel paradigm which aims at revolutionizing the coding development moving from a monolithic application to the composition of multiple services. Micro-services can be deployed independently of one another and are loosely coupled. Each of these micro-services focuses on completing a specific task and interacts with other micro-services by using language-neutral interfaces, such as REST. The REST APIs use HTTP interaction to perform operations on resources and need to be stateless, because the requests might travel through layered intermediaries between the original client and the server. This involves that every request should contain all the information to properly interpret and process that request.

Stateless approaches are testified by several research and industrial solutions. In particular, Mobile Cloud Computing (MCC) [13] [14] promotes the splitting of mobile applications to offload compute and storage intensive tasks to available cloudlet, as long as the latency requirement is preserved. This allows users to achieve better performance and extend the battery life of their personal devices. In [15], a framework to design elastic and scalable edge-based mobile applications has highlighted that most of the components are stateless and their adoption is strongly recommended. Furthermore, the extremely low response time enabled by MEC environments could notably enhance the user experience also for advanced personal cloud storage solutions and user multimedia applications. These applications are typically referred to as stateless since the server is agnostic of the client session state. Therefore, if the client application on the mobile host implements features to recover from server lost connection, without leveraging on session state persistence at networking layers, then it will be able to natively benefit from stateless solutions. Otherwise, specific proxy server could be introduced to transparently preserve the network connection in case of migration [16].

Instead of leveraging only information provided by the client, which could increase the size of the request, stateless applications can also show the ability to store their state in a replicated distributed storage. In this regard, there is an increasing trend to develop stateless network functions [17] [18], which achieve more seamless elasticity and better tolerate failures for the individual devices, while presenting acceptable performance.

B. Service replication management

The challenges related to service replication have been well investigated in the literature for VMs to support high availability (HA). This feature implies the ability to perform continuous failure detection, retrieve and save the state of a VM, and to recover by relying on an existing replica. In particular, to guarantee service continuity, the secondary replica must be tightly coupled and consistent with the primary, such that in case of failure, the replica is always ready to take over without service interruption and data loss.

The approaches used to implement HA for VM can be classified into two categories: (i) record-and-replay and (ii) check-pointing. The former foresees the design of specialized hypervisor which records all input data in the primary VM, sends it over a dedicated link to the secondary replica, and then replay it in the replica [19]. However, to correctly reproduce machine state, this method requires determinism, which could become extremely complex for multiprocessor systems, running over commodity hardware. On the other hand, check-pointing relies on the saving of the whole VM state after the inputs happen, send it to the replica, and keeps the replica VM consistently synchronized with the primary. Remus [20] follows this approach, by implementing a high availability system on top of Xen hypervisor for data center. SecondSite [21] extends this solution to multi-cloud environment, by allowing groups of VMs to be replicated across data centers over wide-area Internet links. However, the performance of this approach heavily depends on the check-pointing frequencies and high amount of data need to be transferred to the replica side. Furthermore, CloudSpider [22] proposes combining VM replication with VM scheduling to reduce migration latencies due to the transfer of large VM image over low bandwidth WAN (Wide Area Network) links. Differently, our approach is focused on stateless micro-services and container-based virtualization, which cope better with the limitation of resource-constrained edge nodes.

Due to the growing interest of lightweight virtualization technologies, remarkable efforts have been addressed to enable high availability also for containers. However, only system-level container, such as OpenVZ and LXC, currently support features of live migration and checkpoint/restore [23]. By relying on this functionality, in [24] a system to support high availability for containers has been developed. However, due to the long duration of check-pointing operations, this mechanism is suitable only for delay tolerant applications. With regard to application containers, such as Docker, the Kubernetes

orchestration tool [25] allows to manage stateless replication by distributing service instance among multiple nodes of the cluster. However, the framework has not been designed for MEC environment and to support service relocation according to user mobility.

III. Service Provisioning in MEC Scenario

In this section, we describe the investigated approaches to support service provisioning and continuity in distributed cloud edge nodes, by highlighting requirements and challenges.

A. Reactive Migration

A first classic approach foresees the migration of a service instance between edge nodes according to user mobility. In particular, when a user moves out from the coverage area of a serving cloudlet, the MEC framework needs to first locate the most suitable node (i.e., both in terms of geographical proximity and resource availability) to accommodate the migrating service. Once the node has been selected, the procedure of migration (Fig. 1) requires:

1. Creating a new service instance at the destination node;
2. Moving persistent data to the new application instance (possibly before starting the instance);
3. Switching user traffic to the new instance;
4. Turning down the original application instance.

Different parameters impact the efficiency of service migration and determine its suitability for specific application scenario. The main relevant metrics are: (*i*) *Downtime*, i.e., the time during which the migrating instance is not running and the user experiences service interruption; (*ii*) *Total Migration Time*, which refers to the overall time for performing the whole migration procedure and includes preparation phase, downtime and resume phase; (*iii*) *Amount of transferred data*, which accounts for both memory and disk data transfer and is fundamental to evaluate the network requirements between hosting nodes.

Accounting for the strict requirements of latency and user mobility, in MEC environment, service continuity is not only related to downtime, but the total migration time also plays a key role to guarantee the user's desired Quality of Service (QoS). Indeed, before the completion of the migration process, the user needs to interact with the instance running in the original serving edge node and the relevant roundtrip time could exceed the tolerable delay. Therefore, the procedure of service migration should be performed within extremely short time intervals, so to preserve low end-to-end latency.

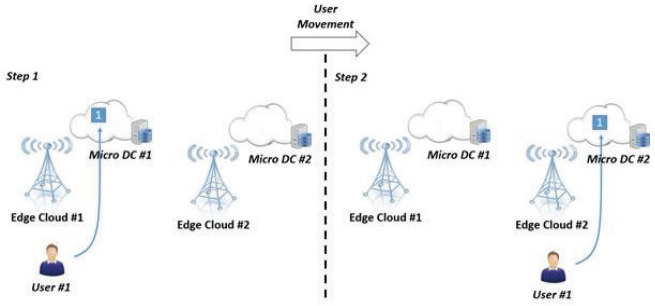

Fig. 1. Example of a reactive service migration in a MEC environment.

B. Proactive Migration

This approach foresees to proactively deploy multiple instances of the user service in neighboring edge nodes. In this way, the long migration time required by service migration could be drastically reduced. In particular, when the event of user movement from a specific service zone is detected, the system needs to select which of the existing replicas is most suitable for the user. The procedure involves the transition of the current serving instance from primary to secondary state, the corresponding transition of another service replica from secondary to primary, and the switching of user traffic from the original replica to the destination replica. Besides, the user movement to the new serving area could trigger the need to relocate some of the remaining existing replicas accordingly.

This approach is also motivated by the reduced overhead deriving from the container-based virtualization, which allows for having high density of service deployment in comparison to VM-based solutions. In this way, the additional cost related for overprovisioning service replicas to guarantee fast response time could be acceptable for delay-sensitive applications in MEC environments.

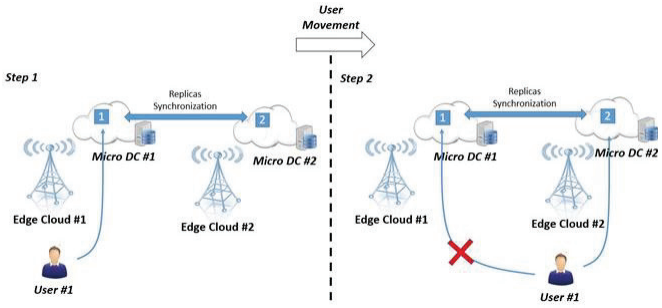

Fig. 2. Example of a proactive service migration approach in a MEC environment by exploiting service replication.

In case of a storage-dependent application, the different replicas should have immediate access to up-to-date stored information. Therefore, the system has to provide a data synchronization system, which manages the periodical propagation of data storage, in order to keep data volume coherency. Accounting also that edge nodes are typically inter-connected with low-bandwidth high-latency links, the data replication mechanism should properly select the update frequency to minimize data traffic and avoid network congestion.

IV. Design Overview

The proposed framework aims to support low latency service provisioning by deploying container-based instances at the network edge. The approach relies on maintaining replicas of specific services to reduce the overall service migration in case of user mobility. This involves the proactively activation of service instances in multiple edge nodes and, for storage-dependent applications, the synchronization of the data among the different replicas. As sketched in Fig. 3, we define the fundamental components and relevant functionalities of our framework. The proposed core functionalities could be also integrated with the ETSI MEC architecture [2].

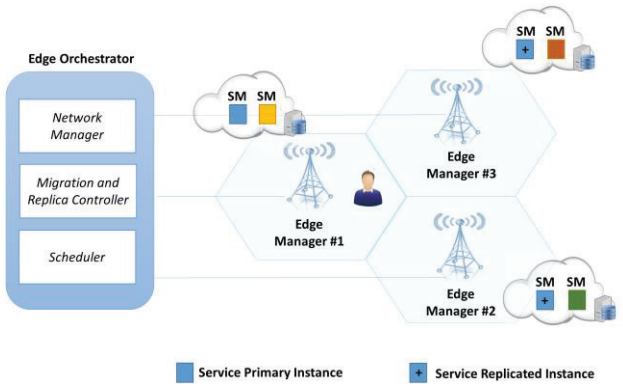

Fig. 3. Architecture of the replication-based MEC framework.

Service Manager (SM): this module constantly monitors the run-time execution of the associated application. Based on the predefined Service Level Agreement (SLA), SM guarantees that sufficient computational and storage resources are allocated to the active replicas, by sending periodical status update to the relevant edge. Besides, this module verifies that existing replicas are coherently synchronized.

Edge Manager (EM): EM is responsible for the management and placement of containers inside a specific cloudlet site. Accounting for the workload and status requirements of the available nodes, it decides how to deploy service instances inside the edge cluster. EM could directly interact with the SMs which are in charge of the services deployed in the edge. Different service criteria could be defined, such as load balancing or energy consumption.

Edge Orchestrator (EO): EO plays a key role in the overall framework by orchestrating the service provisioning over the distributed cloud infrastructure. It comprises the following components:

1. Scheduler: it is responsible for the efficient selection of service replicas and their relevant locations. In particular, the scheduler is continuously updated with the users' location (i.e., access points users connect to) and the edge nodes status, and verifies that current service provisioning is coherent with the predefined SLA. Otherwise, it computes the number of replicas to be deployed for matching application requirements and selects the optimal hosting edge nodes.
2. Network Manager: this module interfaces with the network infrastructure to receive real-time information regarding users' location. When a service migration is required, the existing connections must be preserved to guarantee service continuity. The issue of service migration over WAN is a well investigated topic in the literature and some solutions, such as [26], could be adopted. Moreover, the module needs to carefully check and update the status of connections between federated nodes, in order to guarantee the proper data propagation between primary and relevant replicas.
3. Migration and Replica Controller: This module manages the deployment of the replicas between the different cloudlet sites, based on the decisions provided by the scheduler. The module guarantees that only one of the replicas is active and it acts as primary. Among the remaining replicas, one is also labeled as backup copy for fault tolerance purpose. The state propagation from the primary to the secondary replicas could be performed in synchronous/asynchronous mode, according to the implemented replication technology. When a service migration is requested, this module interacts with the involved EMs to issue the service migration to a different replica. After this process, the flows of state propagation among the replicas need to be updated, modifying the role of primary/secondary role. Moreover, according to the scheduling decision, additional replicas could be instantiated or existing replicas could be relocated between federated edges.

V. Performance Evaluation

A preliminary evaluation of the benefits envisaged by the proposed proactive service replication is provided in this Section. In a system based only on migration strategy, when a user moves to a different access point, he is forced to access the service instance deployed in the previous serving edge node for the *Total Migration Time*, i.e., the time required to activate a new instance in the current edge node. On the other hand, a proactive service replication guarantees no QoE degradation by providing ready replicas in the new serving edge node. Therefore, we aim to assess the time interval to perform service activation in case of a classic service migration in a MEC environment.

In order to estimate the Total Migration Time for stateless applications, we analyze the time required for container startup. In particular, accounting for an application container, the startup time includes the time to pull the relevant image from the service registry (i.e., Docker registry), and the time to launch the container in the edge node. In case of storage-dependent application, the activation time also includes the mounting of additional directories with service files. If these files are not immediately available on the serving node, they are required to be retrieved and migrated from the previous edge node.

The tests have been conducted on a server equipped with an Intel Xeon 3.40 GHz and 8 GB RAM. Docker (version 1.11) is running on top of the operating system (Ubuntu 14.04.3 LTS) allowing the execution of application containers. Each test is repeated 30 times and all the results are shown with a 95% confidence interval. For this preliminary analysis, we select some popular images from the Docker Hub registry and we evaluate the relevant startup time. In Fig. 4, we measure the startup times, in case of precached images and image pulling from the public Docker Hub registry. When images are locally available in the edge node, the startup times present low latency values, around 1 second. However, requesting the container image from the public registry could involve a significant delay. In this case, the service startup time depends on the size of the relevant container image and the connectivity between the edge and the Docker Hub. For large container images, such as a stateless server Httpd and Debian OS respectively of 204 and 125 MBs, the measured startup times are over 15 seconds. Also for small image, like Busybox (2 MBs), the activation time is remarkable and requires around 3 seconds.

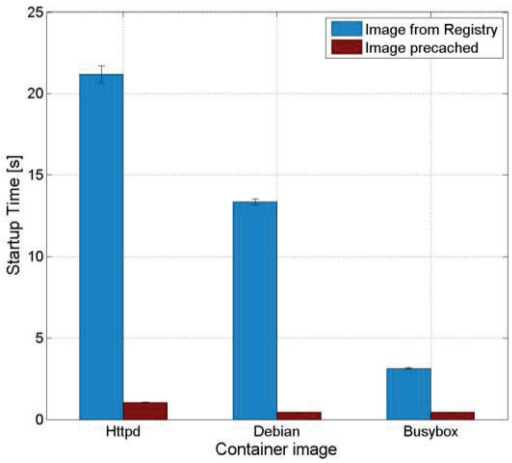

Fig. 4. Container startup time with/out precached images.

In a second analysis, we consider the scenario where container images are not available in public repository, but the container image is exchanged between edge nodes. When an edge node is issued to execute a service, it retrieves the relevant image on-demand from the previous serving edge node. The image is stored and compressed in an archive and, once its transfer is completed, it is loaded locally. For this analysis, we assume another edge node with equivalent hardware and software configuration. In our experiments, we emulate WAN-like conditions using the Linux Traffic Control (tc [27] tool), on physical machines that are connected by Ethernet connection. In particular, we consider a 100 Mbps link for cloud data center interconnectivity, whereas for a MEC environment we configure bandwidth and latency equal to 25 Mbps and 50 ms, respectively, as reported in [28]. The data transfer is performed using scp tool and we use the Httpd container image. The measured values are shown in Fig. 5 and present the different time contributions to: *(i)* transfer image, *(ii)* load in the local repository, and finally *(iii)* launch the relevant containerized service. The main time contribution deals with the image transfer and the overall startup time overcomes 20 and 70 seconds, for both considered configurations.

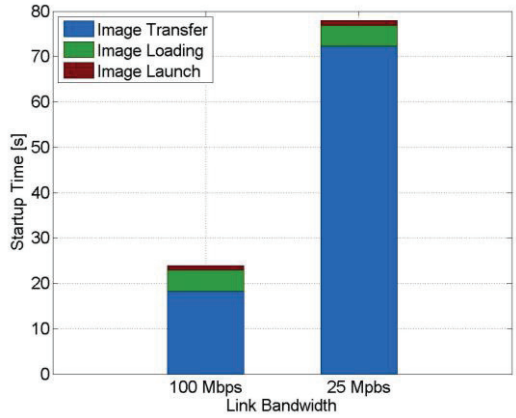

Fig. 5. Httpd container startup with image transfer across edges.

The experimental results highlight that the classic reactive migration approach requires remarkable times to relocate service instances between different edge nodes. Since the traffic forwarding to another edge node increases application response time, the end user can experience QoE degradation during the migration. The integrated use of lightweight virtualization techniques and proactive replication approach seems promising to support next-generation delay-sensitive cloud applications at the network edge, as further investigated in [29].

VI. Challenges and Future Work

The proposed proactive approach raises several research questions and different aspects require further investigation.

A. Replica scheduling policies

To guarantee the desired user QoE, while minimizing the cost deriving by the replication approach, appropriate analytical models should be defined to optimize the number of service replicas. The scheduler of MEC framework should be able to:

(i) Identify the list of eligible cloudlets to host service instances: the list of available edge nodes should be filtered based on both the requirements of the application and the status of nodes. In particular, the framework should carefully evaluate the user perceived latency for service instance hosted on a specific cloudlet. Furthermore, the requirements in terms of processing, storage, and networking should be considered, for both primary and secondary service instances.

(ii) Select the appropriate number of replicas: the framework should carefully evaluate the number of replicas in order to minimize the cost of replication management, while guaranteeing user QoS in case of user location change. This choice could be also dynamical to fit the user mobility and the available network/cloud infrastructure.

(iii) Identify the efficient placement of the replicas at different edge nodes: the selection of the hosting sites is essential to meet application SLA and to minimize the cost of the replication management. This phase could be also enhanced by including prediction of user mobility pattern [30] [31].

B. Data management techniques

Accounting for the limited storage and network capabilities of edge nodes, data management techniques play a remarkable role. As showed by our preliminary results, the availability of container images drastically impacts on the service activation time. Research efforts should be addressed to determine the caching of the more utilized images and to evaluate if the container image layering could be exploited to perform efficient data deduplication and thus to save storage resources. Furthermore, for storage-dependent applications, distributed shared file systems will be required to enable the synchronization of data volumes among different replicas deployed in different edges. In particular, appropriate solutions for data replication should be designed to cope with latency and bandwidth constraints in MEC environment.

C. Support for stateful applications

Finally, the management of stateful applications could extend the possible range of scenarios supported by the proposed framework. However, to support stateful service migration and replication, the underlying container technology should provide appropriate features to manage the state of the hosting applications. To this aim, the efforts performed by the

CRIU project [32] to implement checkpoint features for application containers seem promising. Nonetheless, specific procedures should be designed to propagate the state of the containers among the replicas, while minimizing the overhead of periodical check-pointing. Furthermore, accounting that a new user position can trigger a different deployment of the relevant replicas, the MEC system should be able to perform the migration of several tightly coupled replicas and to keep their coherent synchronization.

VII. Conclusions

In this paper, we have proposed a novel approach for ultra-short latency service provisioning in mobile MEC environments. By leveraging the potential of container technologies and the novel micro-service paradigm, a framework to support proactive service replication for stateless applications has been designed. Accounting for the limitations of current reactive migration strategies, our approach seems promising to support delay-sensitive cloud applications at the network edge, as further investigated in [29]. Future works aim at defining analytical models to optimize proactive service migration among federated edge nodes.

Acknowledgment

This work was partially supported by the TAKE 5 project funded by the Finnish Funding Agency for Technology and Innovation (TEKES) and in part by the Finnish Ministry of Employment and the Economy. It is also partially supported by the European Union's Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement No. 723172.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing,* 2009.

[2] "ETSI GS MEC 003 Mobile Edge Computing: framework and reference architecture v1.1.1," *ETSI MEC ISG, Tech. Rep.,* 2016.

[3] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing ACM*, 2012.

[4] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera and G. Morabito, "Social Virtual Objects in the Edge Cloud," *IEEE Cloud Computing,* 2015.

[5] C. Pahl and B. Lee, "Containers and Clusters for Edge Cloud Architectures--A Technology Review," in *IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*, 2015.

[6] T. Taleb, A. Ksentini, and R. Jantti, "Anything as a Service for 5G Mobile Systems", in IEEE Network, Vol. 30, No. 6, Dec. 2016.

[7] P. Frangoudis, L. Yala, A. Ksentini and T. Taleb, "An architecture for on-demand service deployment over a telco CDN," in *IEEE ICC'16*, Kuala Lumpur, Malaysia, 2016.

[8] S. Dutta, T. Taleb, P. A. Frangoudis, and A. Ksentini, "On-the-fly QoE-Aware Transcoding in the Mobile Edge," in Proc. *IEEE Globecom 2016*, Washington, USA, Dec. 2016.

[9] R. Morabito, J. Kjallman and M. Komu, "Hypervisors vs. Lightweight Virtualization: a Performance Comparison," in *IEEE International Conference on Cloud Engineering*, 2015.

[10] R. Petrolo, R. Morabito, V. Loscrì and N. Mitton, "The design of the gateway for the Cloud of Things," *Annals of Telecommunications,* 2015.

[11] H. Zhuang, R. Rahman and K. Aberer, "Decentralizing the cloud: How can small data centers cooperate?," in *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2014.

[12] Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservice Approach, 2015.

[13] A. R. Khan, M. Othman, S. A. Madani and S. U. Khan, "A survey of mobile cloud computing application models.," *IEEE Communication Surveys & Tutorials,* 2014.

[14] K. Ha, P. Pillai, W. Richter, Y. Abe and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. of the 11th annual int. conference on Mobile systems, applications, and services, ACM*, 2013.

[15] G. Orsini, D. Bade and W. Lamersdorf, "Computing at the mobile edge: Designing elastic android applications for computation offloading".

[16] W. Song, G. Hampel, A. Rana, T. Klein and H. Schulzrinne, "Mosaic: Stateless mobility for http-based applications," in *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2012.

[17] M. Kablan, B. Caldwell, R. Han, H. Jamjoom and E. Keller, "Stateless Network Functions.," in *2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015.

[18] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis and T. Magedanz, "EASE: EPC as a Service to Ease Mobile Core Network," *IEEE Network Magazine,* vol. 29, no. 2, p. 78 – 88, 2015.

[19] D. J. Scales, M. Nelson and G. Venkitachalam, "The design and evaluation of a practical system for fault-tolerant virtual machines. Technical Report VMWare-RT-2010-001, VMWare.," 2010.

[20] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[21] S. Rajagopalan, B. Cully, R. O'Connor and A. Warfield, "SecondSite: disaster tolerance as a service," *ACM SIGPLAN Notices (Vol. 47, No. 7, pp. 97-108),* 2012.

[22] S. K. Bose, S. Brock, R. Skeoch and S. Rao, "CloudSpider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2013.

[23] T. Taleb, S. Dutta, A. Ksentini, I. Muddesar, and H. Flinck "Mobile Edge Computing Potential in Making Cities Smarter," in *IEEE Communications Magazine,* Mar. 2017.

[24] W. Li, A. Kanso and A. Gherbi, "Leveraging linux containers to achieve high availability for cloud services," in *IEEE International Conference on Cloud Engineering*, 2015.

[25] "Kubernetes.," [Online]. Available: http://kubernetes.io/.

[26] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Federated Clouds & Distributed Mobile Networks", in IEEE Network, Vol. 27, No. 5, Sep./Oct. 2013. pp. 12 – 19.

[27] "Linux Traffic Control," [Online]. Available: http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html.

[28] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai and M. Satyanarayanan, "Adaptive vm handoff across cloudlets," Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.

[29] 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,* 2017.

[30] A. Nadembega, A. Hafid and T. Taleb, "An Integrated Predictive Mobile-Oriented Bandwidth-Reservation Framework to Support Mobile Multimedia Streaming," *IEEE Trans. on Wireless Communications,* vol. 13, no. 12, p. 6863 – 6875, Dec. 2014.

[31] I. Farris, T. Taleb, M. Bagaa and H. Flick, "Optimizing Service Replication for Mobile Delay-sensitive Applications in 5G Edge Network," in *IEEE ICC 2017*.

[32] "CRIU. Checkpoint/Restore In Userspace," [Online]. Available: http://criu.org/.