Farris, I.; Taleb, T.; Flinck, H.; Iera, A.

Providing ultra-short latency to user-centric 5G applications at the mobile network edge

Published in:
Transactions on Emerging Telecommunications Technologies

DOI:
10.1002/ett.3169

Published: 01/04/2018

Document Version
Peer reviewed version

Please cite the original version:
Farris, I., Taleb, T., Flinck, H., & Iera, A. (2018). Providing ultra-short latency to user-centric 5G applications at the mobile network edge. *Transactions on Emerging Telecommunications Technologies*, *29*(4), [e3169]. https://doi.org/10.1002/ett.3169

Farris, I.; Taleb, Tarik; Flinck, H.; Iera, A.

Providing ultra-short latency to user-centric 5G applications at the mobile network edge

Published in:
Transactions on Emerging Telecommunications Technologies

DOI:
10.1002/ett.3169

Published: 01/04/2018

Document Version
Peer reviewed version

Please cite the original version:
Farris, I., Taleb, T., Flinck, H., & Iera, A. (2018). Providing ultra-short latency to user-centric 5G applications at the mobile network edge. Transactions on Emerging Telecommunications Technologies, 29(4), [e3169]. https://doi.org/10.1002/ett.3169

TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2012; **00**:1–14
DOI: 10.1002/ett

RESEARCH ARTICLE

Providing Ultra-Short Latency to User-centric 5G Applications at the Mobile Network Edge

I. Farris [1,2] *, T. Taleb [1,3], H. Flinck [4], and A. Iera [2]

[1] Department of Communications and Networking, School of Electrical Engineering, Aalto University, Finland.
[2] DIIES Department, University "Mediterranea" of Reggio Calabria, Italy.
[3] Sejong University, South Korea.
[4] Nokia Bell Labs, Finland.

ABSTRACT

Mobile Edge Computing (MEC) will play a key role in next-generation mobile networks to extend the range of supported delay-sensitive applications. Furthermore, an increasing attention is paid to provide user-centric services, to better address the strict requirements of novel immersive applications. In this scenario, MEC solutions need to efficiently cope with user mobility, which requires fast relocation of service instances to guarantee the desired Quality of Experience. However, service migration is still an open issue, especially for resource-constrained edge nodes interconnected by high-latency and low-bandwidth links. In this paper, by leveraging the potential of lightweight container-based virtualization techniques, we investigate a novel approach to support service provisioning in dynamic MEC environments. In particular, we present a framework where proactive service replication for stateless applications is exploited to drastically reduce the time of service migration between different cloudlets and to meet the latency requirements. The performance evaluation shows promising results of our approach with respect to classic reactive service migration. Copyright © 2012 John Wiley & Sons, Ltd.

***Correspondence**

I. Farris, Department of Communications and Networking, School of Electrical Engineering, Aalto University, Finland. E-mail: ivan.farris@aalto.fi

1. INTRODUCTION

Several classes of new applications are challenging the current network and cloud infrastructures. As illustrated in a recent report from ITU [1], applications such as tactile internet, mobile gaming, and augmented reality are pushing new requirements, especially in terms of latency. To guarantee the desired Quality of Experience (QoE) for the end-user, not only the network access delay, but also the cloud service processing should be performed within a few milliseconds. To achieve the 1 ms latency dream for the next-generation 5G cloud-enabled services, both network and cloud providers should jointly cooperate and investigate novel approaches. In this regard, 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] [3]. Indeed, the deployment of micro datacentres in the network access points, known as cloudlets [4] [5], can guarantee remarkable benefits in terms of low latency interaction and scalability, by balancing the workload over the distributed edge infrastructure. Furthermore, bringing the cloudlet concept into the IoT scenario results in the definition of the so-called Fog Computing paradigm, which envisages a highly virtualized infrastructure composed of distributed edge nodes to monitor and analyse the most time-sensitive data generated by network connected devices [6] [7] [8].

In a MEC environment, a key challenge is represented by the user mobility, which could cause significant application degradation, even in small scenarios, as reported in [9] [10]. 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 area of a specific cloudlet, then the MEC system should be able to promptly detect the user's movement and relocate the user's application by deploying a new service instance, hosted on a cloudlet closer to the end user.

Furthermore, MEC solutions need to tolerate the constraints of heterogeneous cloudlets in term of networking and computing resources. To face this issue, container-based virtualization is considered one of the most promising solutions for MEC environments [11] [12] [13]. Containers allow for avoiding the overhead due to virtualized hardware requested by hypervisor-based virtualization, thus enabling fast initialization and dense deployment of services, as experimented in [14]. All these features make containers extremely attractive for usage not only in datacentres, but also in edge nodes, such as cloudlets and IoT gateways [15] [16]. Nevertheless, enhanced orchestration solutions are required to fully exploit the potential of containerized services for IoT devices at the edge of the network [17] [18] [19].

In this paper, we investigate a novel approach to better support user mobility for stateless micro-services. Indeed, to achieve better scalability and resiliency, several cloud-based applications are being redesigned to be stateless [20], by saving application state and context data into dedicated storage. According to this architecture, in case of migration the service can properly work if the relevant context data is preserved and the needed state can be accessed on-demand. By exploiting the potential of emergent container-based virtualization techniques, we investigate methods to enable fast relocation of stateless micro-services. In particular, we propose a novel method based on lightweight service replication, to reduce the service migration time in comparison to traditional solutions and to increase service resiliency in case of host failure. A preliminary study of our proposal has been presented in [21].

The novel contributions of this paper can be summarized as follows: (i) we present an ETSI-based MEC framework to support efficient service relocation in mobile edge networks; (ii) we define specific procedures to enable proactive migration of container-based services for stateless applications, by exploiting lightweight service replication; (iii) we present a performance evaluation which shows the possible benefits in comparison to classic reactive migration approaches.

The paper is organized as follows. Section 2 describes the background and use-cases for the proposed solution, whereas Section 3 provides an overview on related research work. In Section 4, we present the design of the proposed MEC framework, defining the core modules and relevant functionalities. Section 5 defines the procedures to support fast relocation and replication of container-based services for stateless application with relevant context data. The performance evaluation is reported in Section 6. Manifold open challenges and research areas are discussed in Section 7 and, finally, conclusions are drawn in Section 8.

2. BACKGROUND AND APPLICATION SCENARIOS

Cloud-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 account for internal stored sessions and, instead, they rely on user inputs or backend storage. The stateless feature presents 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 [22], a novel paradigm which aims at revolutionizing the software 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/HTTP APIs. Stateless approaches are testified by main research and application solutions. In particular, Mobile Cloud Computing (MCC) [23] [24] 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 battery life of their personal devices [25] [26].

Instead of leveraging only information provided by the client, which may increase the size of the request, stateless applications can also store their state in a replicated distributed storage. In this regard, there is an increasing trend to develop stateless network functions [27] [28], which achieve more seamless elasticity and better tolerate failures for the individual devices, while presenting acceptable performance.

Accounting for all the above mentioned aspects, our key observation is that next MEC-enabled applications will face a growing deployment of stateless micro-services to address issues of scalability and resiliency. Also, stateless applications may introduce new opportunities to efficiently face the user mobility issue, guaranteeing fast response time in delay critical scenarios. Since these applications rely on user inputs or backend data, they do not have to be migrated by saving all the process and connection states, which could be extremely heavyweight since the transferred state must include the entire memory footprint of the relevant running VM/container. Recent research efforts have demonstrated that traditional migration schemes of VMs and containers still require not-negligible times, especially for constrained cloudlets interconnected with low bandwidth and high latency links [9] [29], and could cause degraded end-to-end latency during the service relocation. On the other hand, in case of stateless applications, only relevant application data must be provided where needed, so that the service can recover the state, resume the processing and properly manage user requests. In this paper, we investigate and design specific migration solutions for stateless applications, where the application is replicated at the new edge node, by preserving the data storage necessary for its correct elaboration.

Several possible applications may achieve benefits from the investigated approaches:

- Ultra short-latency task offloading: a framework to design elastic and scalable edge-based mobile applications is defined in [30] to overcome the challenges of continuously changing and unreliable MEC environments. In the development of sample applications, most of the components are revealed to be stateless and their adoption is strongly recommended. By leveraging the combination of containers and micro-service architecture, the proposed migration approach can promote the widespread of MEC-based applications, efficiently supporting ultra short-latency interactions and seamless user mobility.

- Sensing and IoT-oriented applications: as the number of personal IoT devices is continuously growing, an even stronger impetus is addressed to the development and provisioning of stateless MEC-based IoT applications able to analyse sensing data and improve user experience, by providing real-time feedback and actuation commands. Standardized IoT devices use connection-less protocols, such as Constrained Application Protocol (CoAP) over UDP [31], to interact with external nodes and exploit a stateless REST-based application model, which fits with the proposed approach.

- Content retrieval and video streaming: a considerable amount of today Internet traffic is represented by content retrieval, such as file transfer and video streaming. These applications are typically referred to as stateless since the server is agnostic of the client session state. The extremely low response time enabled by MEC environments can notably enhance the user experience by supporting advanced personal cloud storage solutions and user multimedia applications. Therefore, if the client application on the mobile device implements features to recover from server lost connection, by not leveraging on session state persistence at networking layers, it will be able to natively benefit from the investigated stateless solutions. Otherwise, specific proxy server could be introduced to transparently preserve network connection in case of migration [32].

3. RELATED WORK AND ENABLING TECHNOLOGIES

3.1. Service migration and replication management

Nowadays, migration of service instances represents an essential functionality in cloud distributed infrastructures. Techniques for live migration of VMs in datacentres may be classified in two major approaches: Pre-Copy and Post-Copy memory migration [33]. In the first phase of Pre-Copy approach, the hypervisor copies all the memory pages from source to destination while the VM is still running in the source. Afterwards, the VM will be stopped at the source, the changed pages will be copied to the destination, and VM will be resumed at the destination. Instead, Post-copy first suspends the migrating VM at the source, copies minimal processor state to the target node, resumes the virtual machine, and begins fetching memory pages over the network from the source. 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 VM 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 transfer and is fundamental to evaluate the network requirements between hosting nodes. Several research works have been also conducted to extend live VM migration over distributed Cloud datacentres. In [9] a VM handoff method has been proposed for MEC environment to tolerate the high variability in bandwidth and computation capacity of cloudlets. Despite the remarkable efforts, the results still present significant values of total migration times, which could cause a notable reduction of QoE in MEC environment, since degraded end-to-end latency persists until the end of the operation. Furthermore, the investigated solutions follow a reactive pattern, i.e., the migration is triggered only when the user moves to a different edge node.

The challenges of service replication have been highly studied in the literature for VMs to support high availability (HA) [34]. This feature requires the ability to perform continuous failure detection, clone the state of a VM, and recover by switching to an existing replica. The approaches used to implement HA for VM can be classified into two categories: (i) *record-and-replay* accounts for specialized hypervisor which records all input data in the primary VM, sends it over a dedicated link to the secondary replica, and then replays it in the replica [35]; (ii) *check-pointing* relies on periodically saving of the whole VM state, sends it to the replica, and consistently keeps the replica VM synchronized with the primary [36] [37]. However, the performance of these solutions heavily depends on the check-pointing frequencies and a high amount of data needs to be transferred to the replica side. Furthermore, CloudSpider [38] 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.

Another relevant area of research deals with Application CDN at the edge [39] [40], which investigates the deployment of web services in different edge nodes, by distributing both service code and dynamic content. Whereas our study addresses real-time replication of stateless services based on single user migration, these research work focus on the data management, since concurrent accesses could corrupt data consistency between the web servers. Indeed, differently from a centralized datacentre, application deployed over WAN could not rely on shared disk storage such as a SAN (storage area network) or NAS (network-attached storage). Efficient mechanism for file system synchronization and replication are required for distributed cloud solutions [41].

3.2. Container Virtualization

Container technologies have received a great attention in the last years, enabling efficient lightweight virtualization. System-level containers, such as OpenVZ and LXC, can be executed simultaneously in the same host, by leveraging separate process spaces, file systems, and network stacks. Even if system-level containers guarantee a reduced overhead with respect to VMs and present live migration capabilities, different recent studies [29] [42] have demonstrated that the total migration times cannot meet the strict requirements of delay-sensitive applications. On the other hand, application-oriented containers have attracted great popularity, promoting the paradigm where each container runs a single specific service. In this regard, Docker [43] has emerged as the predominant technology, by defining user-friendly image formats and run-time environment.

Remarkable efforts have been addressed to enable high availability also for container technologies [44]. By relying on checkpoint functionality, in [45], a system to support high availability for system-level 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, the Kubernetes orchestration tool [46] allows managing stateless replication by distributing service instances among multiple nodes of the cluster. However, this framework is not designed for MEC environments and does not support context-based service relocation according to users' mobility.

4. AN ENHANCED MEC FRAMEWORK TO SUPPORT PROACTIVE MIGRATION

To accelerate the widespread of MEC paradigm and promote interoperability among different vendors' solutions, ETSI MEC group has recently released a reference MEC framework [47], which specifies the functional components and their interfaces, i.e., the reference points. In this Section we briefly present the current ETSI MEC framework and we discuss how container-based virtualization and replication mechanisms can be integrated to support ultra-short latency applications in next 5G networks, thus providing an architectural solution compliant with the current standardization activities.

Indeed, although the ETSI MEC specification includes guidelines to support relocation of MEC applications between different edge nodes, their effective realizations are not defined. Our proposed approach relies on maintaining replicas of specific applications, to reduce the overall migration times according to user mobility. This involves the proactive activation of application instances in multiple edge nodes and, for storage-dependent applications, the synchronization of the data among the different replicas. With reference to Fig. 1, we resume the main components of the ETSI architecture and define the requirements to introduce proactive application migration via replication.

Mobile Edge Host (MEH): this entity includes the virtualization infrastructure and the management functionalities, i.e., Mobile Edge Platform (MEP), to effectively perform the deployment of MEC applications at the edge of the network. In particular, the virtualization infrastructure provides the computing, storage, and network resources for MEC applications. A specific feature of MEH deals with the opportunity to offer MEC services, such as Radio Network Information and Location, which can be exploited by MEC applications. However, the provisioning of MEC services strictly depends on the MEHs and it is out of the scope of this paper; therefore, we interchangeably use the terms application and service to identify MEC applications deployed at the edge. Besides, according to the current ETSI specification, MEC applications are defined as VMs running on top of the virtualization infrastructure. To exploit the potential of container-based virtualization in the network edge [29] [48], the MEH infrastructure is required to be extended and provide support for execution of container instances. Once MEC applications are deployed, the Mp1 reference point allows the local MEP to interface with hosted applications to check their availability, manage their session state and traffic rules, and provide access to persistent storage. Whereas Mp2 is used to instruct data plane of the virtualization infrastructure and to route traffic among applications, the Mp3 interface allows for controlling communication between different MEPs. By appropriately exploiting and enhancing Mp1 and Mp3 reference points, the replication mechanism can

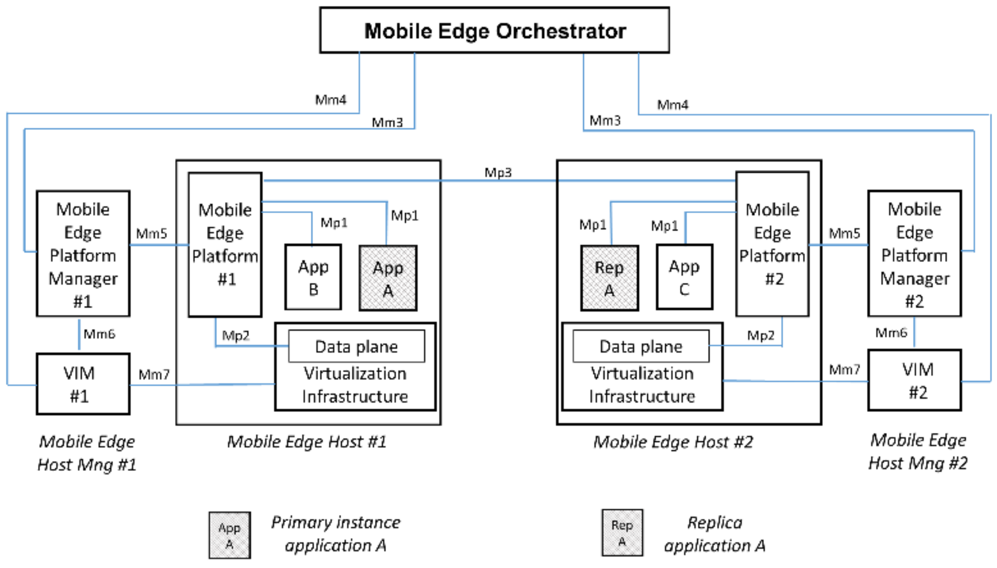


Figure 1. Architecture of the replication-based MEC framework.

be appropriately performed. In particular, state propagation updates are required between primary MEC application and relevant replicas deployed in different MEHs.

Mobile Edge Host Level Management: this section includes both the Virtualization Infrastructure Manager (VIM) and the Mobile Edge Platform Manager (MEPM). The former allocates and manages the resources of the MEH virtualization infrastructure to perform provisioning of applications. Also, it is responsible to provide relevant software images. This represents a key aspect for service relocation of container instances. If the image is not available in the current edge, the VIM needs to pull the requested image from private/public registry, or to transfer the up-to-date image from the previous MEH and to import it in the local catalogue. Furthermore, to guarantee the effective support for ultra-short latency applications, the VIM should implement the proposed procedures of proactive application migration for container-based virtualization, as described in Sec. 5. The MEPM is responsible for monitoring the status of deployed MEC applications and for managing their lifecycles. In case of proactive migration, the MEPM is also required to control the replicas provisioning and constantly verifies their synchronizations.

Mobile Edge Orchestrator (MEO): this module plays a key role in the MEC framework by orchestrating the application provisioning over the distributed edge infrastructure. In particular, it keeps an up-to-date view of the overall mobile edge system, accounting for the status of MEHs, relevant network topology, and user workload. The MEO decides the deployment of applications in the different MEHs based on relevant constraints, such as latency and available resources, and via the relevant VIMs configure the virtualization infrastructure to handle the applications. Furthermore, it periodically validates application requirements and adjust them at run-time. If required, it triggers application relocation between different MEHs. This also requires appropriate network handover between the involved MEHs. The issue of service migration over WAN is a well investigated topic in the literature and some solutions, such as [49] [50], could be adopted. To support our proposed strategy of proactive service relocation, the following functionalities should be introduced in our enhanced MEO:

- Proactive instance scheduling: it is responsible for the efficient selection of application replicas and their relevant locations. In particular, the scheduler is continuously updated with the users' locations (i.e., MEH 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.

- Management of application replicas: based on the scheduling decisions, the MEO is required to orchestrate the deployment of the replicas among the different MEHs. In particular, it must guarantee that only one of the replicas is actively accessed by the user and it acts as primary. Among the remaining secondary replicas, one is also labelled 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 MEHs to issue the service migration to a different replica. Following this process, the roles of primary/secondary instances are updated and the flows of state propagation among the replicas are modified consequently. According to the scheduling decision, either additional replicas could be instantiated or existing replicas could be relocated between federated edges.

5. PROACTIVE APPLICATION MIGRATION IN MEC SCENARIO

To support ultra-short latency applications as those envisioned in Sec. 2, we introduce the concept of proactive migration in a MEC environment. The basic idea is to guarantee fast relocation by deploying multiple instances of the user service in neighbouring edge nodes, so to rely on already available instances whenever handoff to a different cloudlet is necessary. In this way, the long migration time required by reactive service relocation can be drastically reduced. In particular, when the user moves out from the coverage area of a serving cloudlet (hereinafter referred to as source edge), the MEC framework needs to first locate the most suitable node (i.e., in terms of geographical proximity and application requirements) to accommodate the migrating service. Once the target edge is selected, the procedure of migration for context-based stateless application requires that the relevant up-to-date data are locally available to the service, in order to properly access the stored resources and execute its business logic. We underline that the data relevant to the application context should be accessible in the target edge, to reduce possible performance degradation caused by the access to remote shared storages. Different approaches to store context data can be selected according to application requirements, such as relational databases, No-SQL solutions, and object storage. However, to consider a completely Docker container solutions, in this paper we assume the context information are stored in a relevant Data Volume (DV) managed by the edge node.

Indeed, Docker containers are designed to be ephemeral, i.e., all the data generated by a container are lost when the container is removed. To enable data persistence, DVs are introduced to preserve data independently of the container's life cycle. In mobile scenarios, where stateless containers could be executed on-demand in different edge nodes, these DVs represent a core element to enable context persistence. Volumes can be initialized during the creation of Docker container. In addition to creating a volume, also a directory from the hosting node could be mounted by the local Docker engine into a container. Flocker [51] is an open-source container Data Volume Manager which allows moving DVs attached to Docker container between different hosts. However, this solution has been developed for Cloud environments, leveraging block-based shared storage backend. To make DVs available on different potential edge nodes and guarantee extremely fast read/write operation, appropriate mechanisms for data management need to be designed and implemented.

In Fig. 2 we show in detail the procedures required to perform the proactive service migration. After the provisioning of the user application in the MEC system, periodical synchronizations of the relevant DV are carried out between the primary instance and each secondary instance, to keep related application data storage up-to-date. To perform the transmission of data between different edge nodes, the *rsync* [52] utility can be used, which provides fast incremental data transfer. Then, to minimize the migration time and consequently the downtime, we adopt an appropriate pre-copy approach for the migration of the containerized service. Our proposed solution includes the following steps:

1. First DV synchronization: the DV of the source container is transferred to the target edge, to synchronize data with respect to the last synchronization point;
2. Graceful stop of the container: the Docker Engine issues an appropriate signal to smoothly stop the process executed in the container. In this way, on-going user requests or I/O operations can be completed, before shutting down the source container;
3. Final DV synchronization: During the first synchronization, the source container is still running, therefore some files on the target edge server can become outdated. When the container is stopped and its files are not being changed, the second synchronization is performed;
4. Restart of the container on the target server: the container is created and executed on the target server. Accounting that DV with updated context data is mounted on the container, it can gracefully restart;
5. Switching user traffic from the source container to the target container;
6. Releasing the source container, with relevant DV, on the source edge.

An advantage of this procedure lies in its reliability. Indeed, if during the restart phase something goes wrong in the target edge node, the migration procedure can be rolled back, and the container will restart on the source edge, since all data contained in the DV are still locally preserved. Furthermore, if the process executed in the container is also able to update its configuration at run-time, based on the content of the DV, then the container itself might be effectively activated before the migration event (thus, avoiding Step 4 related to container starting). This would contribute to the overall reduction

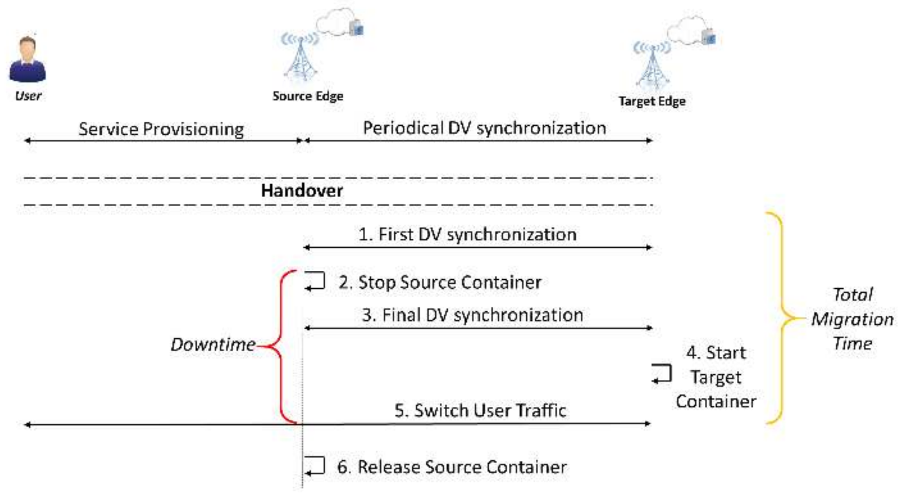

User
Source Edge
Target Edge
Service Provisioning
Periodical DV synchronization
Handover
1. First DV synchronization
2. Stop Source Container
3. Final DV synchronization
Downtime
4. Start Target Container
5. Switch User Traffic
6. Release Source Container
Total Migration Time

Figure 2. Stateless container proactive migration scheme with replication.

of the migration time. Accounting for the periodical data synchronization, the data transfers performed after the trigger of migration procedure will require less time compared to the pure reactive migration solution, since only the differences with respect to the last synchronization point should be managed and a lower amount of data is transferred. This represents a key aspect to effectively support user mobility in MEC environment, where the QoE is strictly dependant not only to the downtime but also to the total migration time. Indeed, until the completion of the migration process, the user needs to interact with the serving instance in the source edge and the relevant round-trip time could exceed the tolerable delay. Therefore, the procedure of service migration shall be performed within extremely short time intervals.

On the other hand, the proposed proactive approach introduces additional costs per each user service, which should be appropriately accounted for. First, the deployment of secondary instance at the edge of the network requires proactive storage of the container image, whose local availability is fundamental to fast instantiate the container in the cooperating edge nodes. In this regard, it is worth underlying that a wise management of application-level containers may reduce the total storage requirements by exploiting similarities among different container images.

In case of a storage-dependent application, the different replicas must also have immediate access to up-to-date stored information. Therefore, the system needs to provide a data synchronization system, which manages the periodical propagation of data updates to maintain DV coherency. This aspect implies that further costs have to be afforded to deploy service instance supported by replication mechanism, in term of storage resources for the DV and bandwidth requirements to guarantee periodical DV synchronization. Edge nodes are typically inter-connected with low-bandwidth and high-latency links; thus, the data replication mechanism shall properly select the update frequency to minimize data traffic and avoid network congestion. Our approach considers a loosely coupled synchronization among the replicas, by leveraging a periodical propagation of the status of the DV content. Instead, a strongly coupled DV synchronization, such as DRBD [53], with synchronous protocol replication may imply a non-negligible application degradation, due to the delay to perform disk operations in the second replica node. To implement asynchronous data replication of DV container, Lsyncd [54] can be exploited to monitor file system changes and periodically perform synchronization; this guarantees a lightweight mirroring solution. Once the migration procedure is successfully performed, the replication framework must also update the role of the different replicas. It changes the state of the previous serving instance from primary to secondary and, correspondingly, the state of the new service replica from secondary to primary. Besides, the user movement to the new serving area could trigger the need to relocate some of the remaining existing replicas accordingly.

During the deployment of replicas, the MEC orchestrator must also reserve enough computation, memory, and networking resources to guarantee the execution of the application adhering to the Service Level Agreement (SLA). Although such a solution could result prohibitive in case of VM-based virtualization, containers present reduced overhead. This feature allows for a high density of service deployments and makes the solution acceptable even in resource-limited cloudlets deployed at the edge of the network. Furthermore, an optimized planning of service replicas, by carefully selecting the number and location in a cellular environment, drastically reduces the

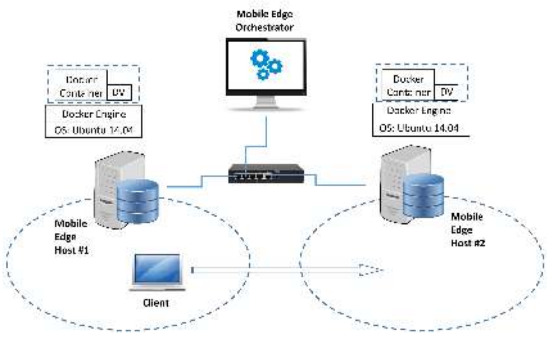


Figure 3. Setup testbed environment.

overall cost. To this aim, in Sec. 7, manifold research challenges are drawn, by identifying possible optimizations to be further investigated.

6. PERFORMANCE EVALUATION

6.1. Testbed description

In this Section we assess the benefits of the proposed proactive service replication mechanism. For this analysis, we build a testbed environment to emulate a multi-edge scenario. In particular, the proactive migration approach is evaluated between two workstations which are representative of the edge clouds. Both workstations use the same operating system Ubuntu 14.04.3 LTS and are running a Docker Engine (version 1.10.3) to provide the virtualization platform for running the backend cloud services inside Docker containers. One server is equipped with an Intel Xeon E3-1231 3.40 GHz and 8 GB RAM, whereas the other is equipped with an Intel Core i5-3472U 1.80 GHz and 8 GB RAM. Another workstation operates as a Mobile Edge Orchestrator to manage the service provisioning in the edge system; in particular, the orchestrator issues commands related to the instantiation, migration, and removal of containerized services in the edge nodes. The proposed analysis does not address the network handover between geo-distributed clouds, which represents a research area complementary to our work and several solutions have been proposed in the literature [49]. Therefore, in our environment, all the servers are connected to the same LAN through an Ethernet switch. The user application, which can interact with the backend service running in the edge system, is executed on a mobile laptop. When a migration event is performed, the IP associated to the backend service container is preserved by properly creating virtual interfaces in the different edge nodes, so that the migration is transparent to the user. The setup of our testbed environment is sketched in Fig. 3.

To emulate a distributed edge environment, we analyse the migration procedure by varying the latencies between the edge nodes. The selected latency values are coherent with the averages values reported in [9]. Latencies between edge nodes are emulated by using the Netem tool, which allows for introducing delay and reproducing MEC conditions in laboratory environments. The bandwidth between edge nodes is equal to 100 Mbps, whereas the latency is increased up to 40 ms.

To better evaluate the benefits introduced by our approach, we consider a traditional reactive stateless migration as benchmark solution for comparison. In literature, some approaches have been investigated for the migration of system-level containers based on a pre-copy approach [29] [42]. To perform a coherent analysis with our proposed solution, we evaluate a similar reactive solution for stateless Docker containerized applications, assuming that the migration is triggered only when the user moves under the coverage of a different edge node. To perform the transmission of data between different edge nodes, we use the *rsync* utility [52] which provides fast incremental data transfer. For our proposed proactive solution, we also adopt the Lsyncd tool [54] to monitor changes in the container DV content and to periodically perform DV synchronization.

6.2. Experimental results

Our analysis aims at assessing the benefits of the proactive solutions, accounting for a realistic scenario, i.e., when the context of a containerized service instance is dynamic and can be modified according to either business logic or user activities. To this aim, we consider a generic-purpose stateless application, so that the results can be easily generalized to different use cases, according to the different service requirements. In particular, to evaluate the impact of data generation, we build a Docker image based on Ubuntu distribution, which generates dynamic random data with varying rates by using the Linux dd tool. The generated files are stored into the relevant DV container, which could be either migrated or replicated across the different edge nodes. For both reactive and proactive migration approaches, the delays related to network handover are not accounted in our testbed environment. We assume different DV sizes, ranging from 10 to 50 MBs. Each service migration test is repeated 30 times and all the results are shown with a 95% confidence interval.

As reported in Fig. 4, the proactive replication-based approach enables shorter migration times since a reduced amount of data is required to be synchronized during the migration event. We achieve a reduction of total migration time equal to 52% and 84% respectively for 10 MBs and 50 MBs in case of 1 ms inter-edge latency, while the relevant benefits range from 25% to 70% in case of 40 ms inter-edge latency. Furthermore, the proactive approach exhibits constant migration times regardless the sizes of the migrated DVs. This makes the solution more scalable and timely efficient for delay-sensitive cloud applications, since only the differences with respect to the last synchronization point should be managed and a lower amount of data is transferred. Furthermore, we underline

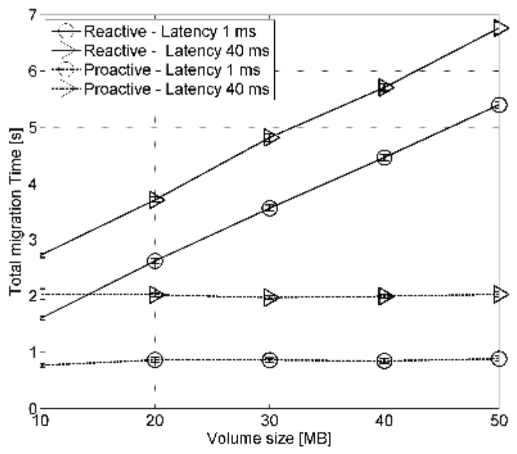


Figure 4. Comparison of total migration time for reactive and proactive service migration approaches.

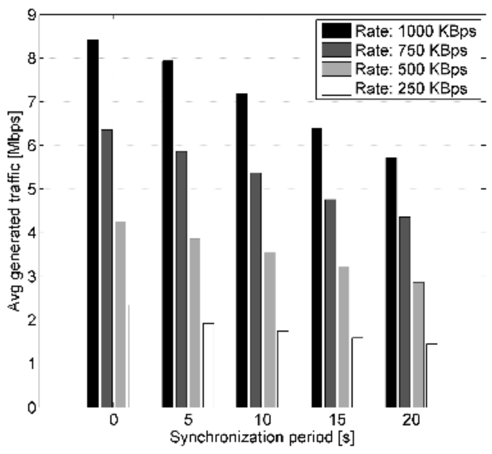


Figure 6. Average generated traffic for proactive approach with different synchronization periods.

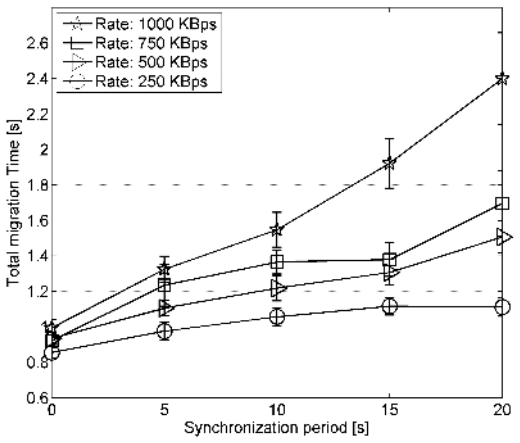


Figure 5. Total migration time for proactive approach with different synchronization periods.

that Docker container images are cached in the edge nodes for both reactive and proactive migration approaches in our analysis. Otherwise, the reactive migration solution would incur an additional delay due to the time necessary to pull the relevant image from a service registry (i.e., a Docker registry). Image data are sent over the network compressed and, then, saved in the local storage uncompressed. The pull time mainly depends on the image size (for example, the size of the used Ubuntu Docker image is equal to 188 MBs).

A key factor in the replication-based approach is the time interval between two subsequent DV content synchronizations, the so-called synchronization period. To evaluate the impact of this parameter on the performance, Fig. 5 shows the resulting migration times for different synchronization periods. In particular, we use the same Docker image to generate new random data stored in the DV with different rates. Before a random migration event within the interval [0 , 60] sec. is simulated, the system is stabilized to guarantee a correct behaviour of the DV replication mechanism. By increasing the time interval between subsequent DV synchronization events in the Lsyncd configuration, the total migration time increases.

To also evaluate the costs of the proposed solution in terms of inter-edge bandwidth requirements, the traffic generated between the primary instance and the replica over 60 seconds is observed (the system is analysed assuring that DV replication mechanism is in a steady state, i.e., the initial synchronization has been previously completed). In case of zero delay between synchronization, the traffic is constant and proportional to the rate of generated data, while in the other cases the traffic presents periodical spikes according to the selected synchronization period. In Fig. 6, the average network traffic between primary and replica assumes lower values when decreasing the synchronization period, since Lsyncd monitors content in the DV over the selected period, aggregates and combines the detected data changes, and then synchronizes the replica.

Another relevant parameter to evaluate the feasibility of the proposed replication approach in an ultra-dense cellular system is represented by the initialization time, i.e., the time to deploy a replica in a secondary edge node. Indeed, to increase the capabilities of the cellular network, next-generation 5G system [55] will envision the coexistence of heterogeneous small cells, such as femtocells and WiFi access points, with different coverage areas, also able to host cloud services at the edge. In Fig. 7, we plot the values required to initialize a replica of the considered container by accounting for different DV sizes and latency values between the edge nodes. According to the initialization times and edge node coverage, the

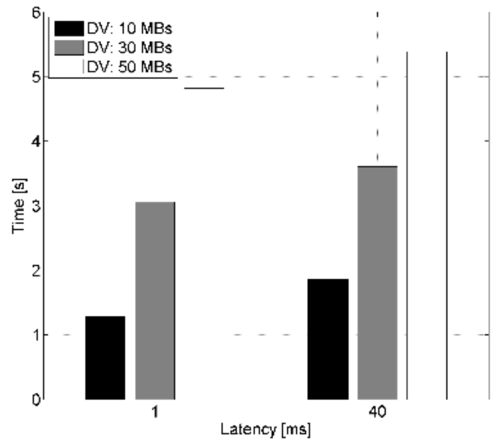


Figure 7. Average initialization time to activate a new replica in a different edge-node.

replication-based scheme introduces constraints on the maximum velocity of the mobile users. For instance, assuming a femtocell radius ranging from 20 to 50 meters [56] and the linear crossing of the user over the maximum cell extension, the initialization time in case of DV size of 50 MBs implies a user mobility below 26 and 67 km/h respectively, to guarantee the effective deployment of the replica. It is worth underlying that these speed values are coherent, in the former case, with users with low mobility (such as pedestrians and cyclists) and, in the latter case, with user moving on vehicle within urban environments (which usually set specific speed limits). This analysis provides useful indications for the MEC orchestrator to appropriately select edge nodes for replica deployment.

7. CHALLENGES AND FUTURE WORKS

The integrated use of lightweight virtualization techniques and proactive replication approach seems promising to support next-generation delay-sensitive cloud applications. Moreover, the proposed proactive approach raises several research issues that require further investigation.

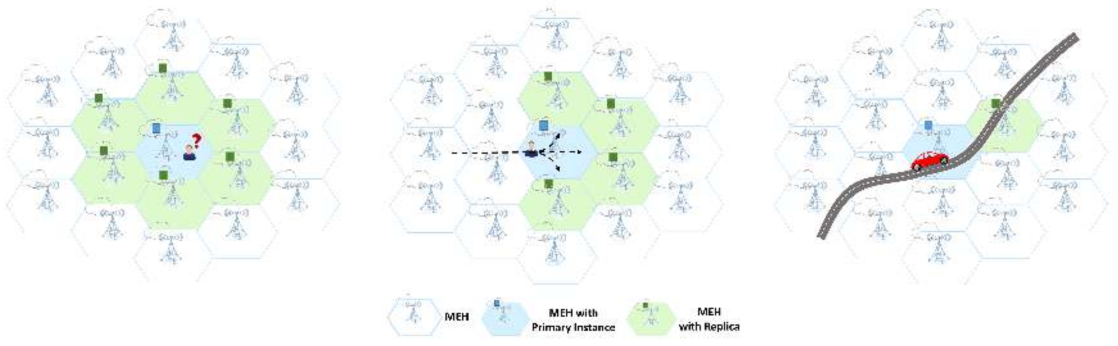
7.1. Replica Scheduling policies

To guarantee the desired user QoE, while minimizing the costs of replication approach, appropriate analytic models should be defined to optimize the number of service replicas and the adopted deployment policy. In particular, the MEC orchestrator has to:

- identify the list of eligible cloudlets to host service instances: in this phase, 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, as well as the network bandwidth and latency between different edge nodes. Furthermore, the framework should consider the requirement in terms of processing and storage required by a specific service, both for primary and secondary service instances.
- select the appropriate number of replicas: the framework should carefully evaluate the number of replicas to minimize the cost of replication management, while guaranteeing user QoS in case of user location change. This choice might change dynamically to fit the user mobility and the available network/cloud infrastructure.
- 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 can be also enhanced by estimating edge node workloads, wireless connectivity and exploiting the prediction of user mobility patterns [57] [58].

Indeed, by considering a traditional cellular cluster topology, a naive approach is represented by distributing replicas in all the near edges to the serving cloudlet, as sketched in Fig. 8 (a). However, this approach involves two drawbacks. On the one hand, there is a risk of underutilizing reserved resources in edge nodes which the user will never connect to, while, on the other hand, the prospect is to overload cloudlets with replicas if a large number of user requires delay-sensitive services (thus reducing the number of new effective services deployed due to lack of available resources). As a possible solution to reduce the impact of service replication, precise estimation of user mobility pattern can drastically reduce the cost of replica deployment. For instance, by exploiting information about user position and orientation, the system could perform simple predictions of future user path and replicate services only at the target edges towards the user mobility direction. In this way, the number of involved edge nodes may be almost halved, as depicted in Fig. 8 (b). Similarly, in traditional cellular systems, proactive channel reservation schemes and relevant analytical models have been highly investigated to reduce the probability of dropped calls during user handovers by exploiting user mobility prediction [59]. To this aim, a group of channels is reserved to ensure the priority of successful handover calls. However, these available channels can be interchangeably exploited by all the users moving to the same cell. This differs from our MEC scenario where each deployed application instance is strictly associated to a single user, accounting for the relevant context data. Therefore, the replicas deployed in the near edge nodes are reserved only for the user which has requested the related service.

More accurate prediction of user path can be considered by building a mobile service usage cartography based on filtered historical movement pattern and



(a) (b) (c)

Figure 8. Examples of replicas deployment in a cellular network with naive approach (a), user direction prediction (b), and vehicular mobility prediction (c).

contextual knowledge [60]. For instance, in vehicular urban environment, highly accurate estimation of handoff events along the path to destination can be modelled by accounting for the time taken to transit road segments along the path, navigation zone characteristics, and behaviour of users on specific road segments. In this way, the cost of service replication could be further alleviated by deploying the service replicas only in specific edge nodes leveraging on the predicted user trajectory, as shown in Fig. 8 (c). An initial study on optimization of replica deployment for mobile users in edge environments has been presented in [61]. Furthermore, by using models able to predict the user path from source to destination [62], the MEC system could evaluate if enough resources can be reserved in the cloudlets along the user route, to plan extremely fast migrations between the selected edge nodes and guarantee the desired QoE.

7.2. Support for stateful applications

Finally, the management of stateful applications can 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 states of the hosting applications. To this aim, the efforts performed by the CRIU project [63] to implement checkpoint features for Docker containers seem promising.

Nonetheless, specific procedures need to be designed to propagate the states of the containers among the replicas, while minimizing the overhead of periodical check-pointing. Indeed, depending on the frequency of checkpoint replication and the amount of state changed since the previous checkpoint, replication for VM-based virtualization can easily consume up to some hundreds Mbps for memory intensive workloads [36]. These data flows are unmanageable by edge nodes that are typically connected with each other through high latency and low bandwidth links. As a consequence, the overall traffic due to state synchronization among replicas must be reduced to cope with networking resources that are constrained and also quite expensive. Another key issue is the appropriate management of service replicas in case of tight synchronization. When the MEC system detects user movements and decides a different replica attachment for a specific application, then control flows for state propagation among replicas should be appropriately modified to keep their coherent synchronization.

Another emerging approach to enable scalable applications relies on moving application states into backend systems [27]. To provide fast access to on-demand state and guarantee adequate application performance, extremely low-latency interaction between the processing service and relevant data store is required. In case of distributed cloud environment, the backend data system needs to manage the state replication over multiple sites by providing a distribution protocol which guarantees state coherency among replicas. To consider this novel development model, our proposed proactive migration could also be extended by enabling integrated replication of both stateless service and relevant backend system. Future work will be addressed to investigate and evaluate the impact of this paradigm in a MEC environment.

8. CONCLUSIONS

In this paper, we proposed a novel approach for the provisioning of ultra-short latency applications in MEC environments. Accounting for the potential of container technologies and for the novel micro-service paradigm, a framework to support proactive service migration for stateless applications has been designed. The conducted performance evaluation assessed the potential of our solution in guaranteeing fast service relocation within MEC environments. Future research will be conducted to define analytical models to optimize service replication among federated edge nodes, so to guarantee the desired user experience while reducing the relevant costs.

9. 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 Unions Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement No. 723172.

REFERENCES

1. The tactile internet. *Technical Report*, ITU-T Technology Watch Report 2014.
2. ETSI ISG MEC. Mobile-edge computing-introductory technical white paper 2014.
3. Ahmed A, Ahmed E. A survey on mobile edge computing. *the Proceedings of the 10th IEEE International Conference on Intelligent Systems and Control (ISCO 2016), Coimbatore, India*, 2016.
4. Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE* 2009; **8**(4):14–23.
5. Shaukat U, Ahmed E, Anwar Z, Xia F. Cloudlet deployment in local wireless area networks, motivation, taxonomies, and open research challenges. *J. Netw. Comput. Appl* 2016; **62**.
6. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, 2012; 13–16.
7. Mehta A, Trneberg W, Klein C, Tordsson J, Kihl M, Elmroth E. How beneficial are intermediate layer data centers in mobile edge networks? *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016; 222–229, doi:10.1109/FAS-W.2016.55.
8. Farris I, Girau R, Militano L, Nitti M, Atzori L, Iera A, Morabito G. Social virtual objects in the edge cloud. *IEEE Cloud Computing* Nov 2015; **2**(6):20–28, doi:10.1109/MCC.2015.116.
9. Ha K, Abe Y, Chen Z, Hu W, Amos B, Pillai P, Satyanarayanan M. Adaptive vm handoff across cloudlets. *Technical Report*, Technical Report CMU-CS-15-113, CMU School of Computer Science 2015.
10. Ahmed E, Akhunzada A, Whaiduzzaman M, Gani A, Ab Hamid SH, Buyya R. Network-centric performance analysis of runtime application migration in mobile cloud computing. *Simulation Modelling Practice and Theory* 2015; **50**:42–56.
11. Pahl C, Lee B. Containers and clusters for edge cloud architectures–a technology review. *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, IEEE, 2015; 379–386.
12. Taleb T, Ksentini A, Jantti R. ”anything as a service” for 5g mobile systems. *IEEE Network* November 2016; **30**(6):84–91.
13. Frangoudis P, Yala L, Ksentini A, Taleb T. An architecture for on-demand service deployment over a telco cdn. *IEEE ICC 16,*, IEEE, 2016.
14. Morabito R, Kjallman J, Komu M. Hypervisors vs. lightweight virtualization: a performance comparison. *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, IEEE, 2015; 386–393.
15. Ahmed E, Yaqoob I, Gani A, Imran M, Guizani M. Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. *IEEE Wireless Communications* October 2016; **23**(5):10–16, doi:10.1109/MWC.2016.7721736.
16. Petrolo R, Morabito R, Loscrì V, Mitton N. The design of the gateway for the cloud of things. *Annals of Telecommunications* 2017; **72**(1):31–40, doi:10.1007/s12243-016-0521-z. URL http://dx.doi.org/10.1007/s12243-016-0521-z.
17. Hegyi A, Flinck H, Ketyko I, Kuure P, Nemes C, Pinter L. Application orchestration in mobile edge cloud: Placing of iot applications to the edge. *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016; 230–235.
18. Q, AlBalas F, Jararweh Y, Al-Ayyoub M. A fog computing based system for selective forwarding detection in mobile wireless sensor networks. *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016; 256–262.
19. Li W, Santos I, Delicato FC, Pires PF, Pirmez L, Wei W, Song H, Zomaya A, Khan S. System modelling and performance evaluation of a three-tier cloud of things. *Future Generation Computer Systems* 2016; .
20. Cockroft A, Hicks C, Orzell G. Lessons netflix learned from the aws outage. http://techblog.netflix.com/2011/04/lessons-netflix-learned-from-aws-outage.html. Accessed: November 2016.
21. Farris I, Taleb T, Iera A, Flinck H. Lightweight service replication for ultra-short latency applications in mobile edge networks. *IEEE ICC 2017*, IEEE, 2017.
22. Daya S, Van Duy N, Eati K, Ferreira CM, Glozic D, Gucer V, Gupta M, Joshi S, Lampkin V, Martins M, *et al.*. *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*. IBM Redbooks, 2016.
23. Ahmed E, Gani A, Sookhak M, Ab Hamid SH, Xia F. Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. *Journal of Network and Computer Applications* 2015; **52**:52–68.

24. Ahmed E, Gani A, Khan MK, Buyya R, Khan SU. Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications* 2015; **52**:154–172.
25. Orsini G, Bade D, Lamersdorf W. Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications. *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016; 216–221, doi:10.1109/FAS-W.2016.54.
26. Lo'ai AT, Bakheder W, Song H. A mobile cloud computing model using the cloudlet scheme for big data applications. *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016 IEEE First International Conference on*, IEEE, 2016; 73–77.
27. Kablan M, Caldwell B, Han R, Jamjoom H, Keller E. Stateless network functions. *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ACM, 2015; 49–54.
28. Taleb T, Corici M, Parada C, Jamakovic A, Ruffino S, Karagiannis G, Magedanz T. Ease: Epc as a service to ease mobile core network deployment over cloud. *Network, IEEE* 2015; **29**(2):78–88.
29. Taleb T, Dutta S, Ksentini A, Muddesar I, Flinck H. Mobile edge computing potential in making cities smarter. *IEEE Communications Magazine* 2017; .
30. Orsini G, Bade D, Lamersdorf W. Computing at the mobile edge: Designing elastic android applications for computation offloading. *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, IEEE, 2015; 112–119.
31. Palattella MR, Accettura N, Vilajosana X, Watteyne T, Grieco LA, Boggia G, Dohler M. Standardized protocol stack for the internet of (important) things. *Communications Surveys & Tutorials, IEEE* 2013; **15**(3):1389–1406.
32. Song W, Hampel G, Rana A, Klein T, Schulzrinne H. Mosaic: Stateless mobility for http-based applications. *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, IEEE, 2012; 69–76.
33. Ahmad RW, Gani A, Hamid SHA, Shiraz M, Yousafzai A, Xia F. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications* 2015; **52**:11–25.
34. Colman-Meixner C, Develder C, Tornatore M, Mukherjee B. A survey on resiliency techniques in cloud computing infrastructures and applications. *IEEE Communications Surveys Tutorials* 2016; **PP**(99):1–1, doi:10.1109/COMST.2016.2531104.
35. Scales DJ, Nelson M, Venkitachalam G. The design of a practical system for fault-tolerant virtual machines. *ACM SIGOPS Operating Systems Review* 2010; **44**(4):30–39.
36. Cully B, Lefebvre G, Meyer D, Feeley M, Hutchinson N, Warfield A. Remus: High availability via asynchronous virtual machine replication. *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, 2008; 161–174.
37. Rajagopalan S, Cully B, O'Connor R, Warfield A. Secondsite: disaster tolerance as a service. *ACM SIGPLAN Notices*, vol. 47, ACM, 2012; 97–108.
38. Bose SK, Brock S, Skeoch R, Rao S. Cloudspider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks. *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, IEEE, 2011; 13–22.
39. Rabinovich M, Xiao Z, Aggarwal A. Computing on the edge: A platform for replicating internet applications. *Web content caching and distribution*. Springer, 2004; 57–77.
40. Sivasubramanian S, Pierre G, Van Steen M. Replicating web applications on-demand. *Services Computing, 2004.(SCC 2004). Proceedings. 2004 IEEE International Conference on*, IEEE, 2004; 227–236.
41. Suel T, Noel P, Trendafilov D. Improved file synchronization techniques for maintaining large replicated collections over slow networks. *Data Engineering, 2004. Proceedings. 20th International Conference on*, IEEE, 2004; 153–164.
42. Machen A, Wang S, Leung KK, Ko BJ, Salonidis T. Migrating running applications across mobile edge clouds: poster. *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, ACM, 2016; 435–436.
43. Docker container. `https://www.docker.com/`. Accessed: November 2016.
44. Li W, Kanso A. Comparing containers versus virtual machines for achieving high availability. *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, IEEE, 2015; 353–358.
45. Li W, Kanso A, Gherbi A. Leveraging linux containers to achieve high availability for cloud services. *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, IEEE, 2015; 76–83.
46. Kubernetes. `http://kubernetes.io/`. Accessed: November 2016.
47. Etsi gs mec 003 mobile edge computing (mec); framework and reference architecture v1.1.1. *Technical Report*, ETSI MEC ISG 2016.
48. Dutta S, Taleb T, Frangoudis PA, Ksentini A. On-the-fly qoe-aware transcoding in the mobile edge. *IEEE Globecom 2016*, IEEE, 2016.
49. Taleb T, Ksentini A. Follow me cloud: interworking federated clouds and distributed mobile networks. *Network, IEEE* 2013; **27**(5):12–19.

50. Raad P, Secci S, Phung DC, Cianfrani A, Gallard P, Pujolle G. Achieving sub-second downtimes in large-scale virtual machine migrations with lisp. *Network and Service Management, IEEE Transactions on* 2014; **11**(2):133–143.
51. Flocker. https://github.com/ClusterHQ/flocker. Accessed: November 2016.
52. rsync. https://rsync.samba.org/. Accessed: November 2016.
53. Drbd (distributed replicated block device). http://www.drbd.org/. Accessed: November 2016.
54. lsyncd. https://github.com/axkibe/lsyncd. Accessed: November 2016.
55. Ge X, Tu S, Mao G, Wang CX, Han T. 5g ultra-dense cellular networks. *IEEE Wireless Communications* February 2016; **23**(1):72–79, doi:10.1109/MWC.2016.7422408.
56. Chandrasekhar V, Andrews JG, Gatherer A. Femtocell networks: a survey. *IEEE Communications magazine* 2008; **46**(9):59–67.
57. Bagaa M, Taleb T, Ksentini A. Service-aware network function placement for efficient traffic handling in carrier cloud. *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2014; 2402–2407.
58. Xu D, Ren P, Sun L, Song H. Precoder-and-receiver design scheme for multi-user coordinated multi-point in lte-a and fifth generation systems. *IET Communications* 2016; **10**(3):292–299.
59. Sgora A, Vergados DD. Handoff prioritization and decision schemes in wireless cellular networks: a survey. *IEEE Communications Surveys & Tutorials* 2009; **11**(4):57–77.
60. Nadembega A, Taleb T, Hafid A. A destination prediction model based on historical data, contextual knowledge and spatial conceptual maps. *2012 IEEE International Conference on Communications (ICC)*, IEEE, 2012; 1416–1420.
61. Farris I, Taleb T, Bagaa M, Flinck H. Optimizing service replication for mobile delay-sensitive applications in 5g edge network. *IEEE ICC 2017*, IEEE, 2017.
62. Nadembega A, Hafid A, Taleb T. An integrated predictive mobile-oriented bandwidth-reservation framework to support mobile multimedia streaming. *Wireless Communications, IEEE Transactions on* 2014; **13**(12):6863–6875.
63. Criu. checkpoint/restore in userspace. http://criu.org/. Accessed: November 2016.