IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INFORMATION CENTRIC WIRELESS NETWORKING WITH EDGE COMPUTING FOR 5G AND IOT

Received February 21, 2019, accepted March 15, 2019, date of publication April 17, 2019, date of current version April 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907457

Qos-Aware Service Continuity in the Virtualized Edge

HADEEL ABDAH, JOÃO PAULO BARRACA, AND RUI L. AGUIAR
Instituto de Telecomunicacoes, University of Aveiro, 3810-193 Aveiro, Portugal

Corresponding author: Hadeel Abdah (hadeel.abdah@ua.pt)

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER - PT2020 partnership agreement under the project UID/EEA/50008/2019, and Fundação para a Ciencia e Tecnologia under Grant SFRH/BD/136361/2018

ABSTRACT 5G systems are envisioned to support numerous delay-sensitive applications such as the tactile Internet, mobile gaming, and augmented reality. Such applications impose new demands on service providers in terms of the quality of service (QoS) provided to the end-users. Achieving these demands in mobile 5G-enabled networks represent a technical and administrative challenge. One of the solutions proposed is to provide cloud computing capabilities at the edge of the network. In such vision, services are cloudified and encapsulated within the virtual machines or containers placed in cloud hosts at the network access layer. To enable ultrashort processing times and immediate service response, fast instantiation, and migration of service instances between edge nodes are mandatory to cope with the consequences of user's mobility. This paper surveys the techniques proposed for service migration at the edge of the network. We focus on QoS-aware service instantiation and migration approaches, comparing the mechanisms followed and emphasizing their advantages and disadvantages. Then, we highlight the open research challenges still left unhandled.

INDEX TERMS Service migration, service functions virtualization, mobile edge computing, QoS-awareness.

I. INTRODUCTION

With the exponential increase in the number of mobile users, the innovation of new sophisticated real-time services, and the ever-increasing proliferation of smart wireless devices, current cellular wireless networks will have to advance in various ways to fulfill the presumptions and challenges of the near future. It is commonly assumed that the next generation of cellular wireless networks, referred generically as 5G networks, must address six vectors that are not effectively addressed by the currently deployed 4G: higher capacity, higher data rate, lower end-to-end latency, massive device connectivity, reduced cost, and consistent Quality-of-Experience (QoE) provisioning [1], [2]. To meet these requirements and to overcome the new challenges facing 5G systems, a drastic change in the design of cellular infrastructures is needed. One of the changes proposed is to extend cloud computing to the edge of the network, providing resources, services and computing capabilities from close proximity to the end users. This implementation is usually considered to take one of three forms. They are Multi-access Edge Computing (MEC), Cloudlet-Based Paradigm (CBP) and Fog Computing (FC) [3].

Multi-access Edge Computing, previously known as Mobile Edge Computing, moves the computing capabilities of the network to the edge, within the Radio Access Network (RAN) and closer to mobile subscribers [4]. The MEC paradigm is proposed for the upcoming 5G systems, to enable support for delay-sensitive applications such as tactile Internet, mobile gaming, and augmented reality. Such applications push new requirements to service provisioning in terms of QoE for the end-user. In this context, to enable ultrashort processing times and immediate service response, fast instantiation and migration of service instances between edge nodes is mandatory to cope with the consequences of user's mobility.

In the Cloudlet-Based Paradigm, the Cloudlet is defined as ''a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby mobile devices'' [5]. In the Cloudlet-Based

The associate editor coordinating the review of this manuscript and approving it for publication was Khalil Afzal.

mobile computing environment, mobile devices send jobs to the cloudlet, where they are processed and the results are sent back. As a consequence, this reduces transmission delay and power consumption of the mobile device. Therefore, it presents the prospects for a great evolution in Mobile Cloud Computing (MCC) [6].

Fog Computing, same as Multi-access Edge and Cloudlet-Based paradigms, extends the Cloud Computing (CC) paradigm to the edge of the network to enable the support for latency-intolerant applications. However, FC is more intended to support Internet of Things (IoT) devices.

While MEC implementation is associated with servers co-located with a Radio Network Controller (RNC) or a macro base station, the cloudlet is considered to be ''data center in a box'' which the devices connect directly to over Wireless local area networking (WIFI). FC nodes, on the other hand, are heterogeneous and can be routers, switches, access points or gateways, and may be multiple hops away from the User Equipment. In-depth comparisons of the multiple implementations of Virtualized Edge (VE) are extremely relevant, and have been pursued by some authors [3].

All three approaches share the virtualized nature of the services running on them. In addition, they all aim at supporting latency-aware services by getting resources and computing capabilities closer to the users.

MEC has not been standardized yet, with telecom providers being focused in proof of concepts, interoperability and viability studies. In this sense, there have been attempts to integrate MEC and other technologies, such as fiber-wireless access networks [7]. In this case, the use of the MEC paradigm has achieved a delay within 1 ms, as a direct result of eliminating the interactions with the backhaul network components. Furthermore, the results proved that MEC paradigm has significantly better throughput, lesser Round Trip Time (RTT), and lesser packet loss than the MCC paradigm. As it is expected, the long physical distance between communicating components, and the greater number of forwarding hosts in MCC, significantly affect the overall network performance [7] and the perceived QoE.

The Cloudlet-based architecture, also not standardized, was evaluated [8] and proved to provide reduced the access latency and satisfy the users' demands in terms of delay. Through the implementation of FC it was shown that an RTT of 1.416 ms was possible to obtain, compared to 17.989 ms in CC [9]. These works present a bright future for these technologies, enabling the use of future telecom networks in delay sensitive applications.

This paper presents a survey on techniques used to ensure efficient service migration at the networks virtualized edge, regardless of the implementation and naming. Efficiency service migration is a key aspect of FC and MEC, as it will be highly stressed under networks observing high load, or highly mobile users. We focus on Quality-of-Service (QoS)-aware service instantiation and migration approaches, comparing the mechanisms followed, and highlighting their advantages and disadvantages. From here forward, we use the term VE, as a term that encompasses all three concepts Fog Computing, Multi-access Edge Computing, and Cloudlet-Based environments.

Following, in section II, we explain the concept of service migration in general. While section III describes the difficulties facing migration process, in section IV, we explain how QoS is preserved in a Virtualized Edge, focusing on QoS-aware service migration and detailing the methodologies followed, their advantages and disadvantages. The simulation platforms that support Virtual Machine (VM) migration are detailed and compared in section V. The following section VI summarizes the challenges unhandled by the proposed approaches. We then provide some future directions in section VII. The section VIII is dedicated for the conclusions. Finally, a bibliography is in the last section.

II. SERVICE MIGRATION CONCEPT

Services in a virtualized environment are typically encapsulated in virtual machines or containers. A VM running on a first physical host can be migrated to another physical host. The migration process encompasses the transfer of the persistent state of the VM (i.e. its file system), the transfer of the volatile state of the VM (i.e. Random Access Memory (RAM) contents and Central Processing Unit (CPU) state), and the redirection of network traffic [10]. The different phases of the service migration process are depicted in Figure 1. Service migration procedures can be classified into different categories based on different criteria, which we analyze in the following paragraphs.

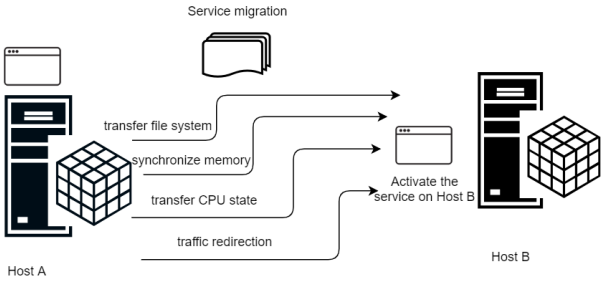

FIGURE 1. Service migration process.

A. BASED ON THE SERVICE TYPE

Service migration can be classified according to the service type into stateful and stateless migration. Stateless migration does not move the application running states, it only redirects the user requests to a new instance of the service, located on another server. On the contrary, stateful migration involves moving running applications together with its execution states (memory pages and kernel context). The former is applicable for applications which do not keep state for users. However, for interactive services, such as active safety warning, mobile multimedia, and mobile online gaming, it is very likely that the application needs to keep some state for each user. Thus, stateful migration is required [11].

B. BASED ON THE SERVICE STATUS DURING THE MIGRATION

We can identify live migrations and cold migrations. i) Live migrations describe the process of copying a VM from one hypervisor host to another hypervisor host, while the VM is still executing [12]. Live migration should be transparent to the guest Operating System (OS), its applications, and to the remote users of the VM [14]. Two predominant techniques exist, which are: pre-copy live migration, and post-copy live migration [15]. The pre-copy approach involves transferring the memory of the VM from a source to a destination through several iterations before the VM is restarted, whilst the post-copy approach only sends the Virtual CPU (vCPU) and the device state to the destination at an initial stage. Subsequent pages are only sent on demand while the VM is running on the destination host [16]. Most virtualization environments support live migration, allowing administrators to move a VM between physical hosts within a Local Area Network (LAN) platform, without greatly disrupting service (e.g., XenMotion, VMware vMotion, KVM Live Migration) [4]. ii) Cold migration describes the process through which the service is paused or terminated while the VM states are transferred. The advantage of this type of migration lies in its low complexity. However, it may incur a significant service downtime, since the VM states can comprise vCPU, memory, disk, and network, requiring the transfer of several Gigabytes of data [12]. The work-flows of Cold, pre-copy and post-copy migrations are illustrated in Figure 2 a, b, and c subsequently.

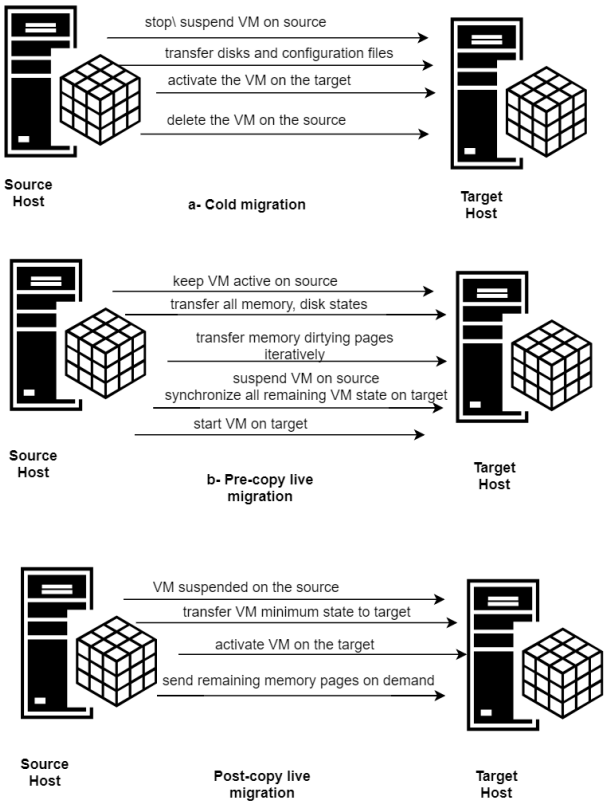

FIGURE 2. Service live and cold migrations.

C. BASED ON THE MIGRATION TRIGGERING TIME

Based on this criteria, we can distinguish two categories, reactive and proactive. This classification is usually solely related to the VE environment. A reactive approach which requires migrating a service instance from one edge node to another following the movement of its user. When a user moves out of the coverage area of a serving edge node, the MEC paradigm needs to first find out the most suitable target edge (i.e., both in terms of geographical proximity and resource availability) to host the service candidate to be migrated. Once the target edge is found, the migration process is initiated.

In proactive approaches, the service replication is performed based on predicting beforehand where the user is heading. Thus, the replica is migrated and deployed in the neighboring edge nodes before the user is in the coverage area of these edges. In this approach, the service is readily available if the handover to the target edge node is necessary. Long migration times required by on-demand service relocation could be drastically reduced, but miss-prediction will waste resources [13].

D. BASED ON THE SERVICE ENCAPSULATION

A service application is often encapsulated into its own self-sufficient and pre-configured environment for easy distribution. Current examples of such an environment are the well established hypervisor-based VMs, or the more recent technique, containers. Both technologies allow the creation and running of multiple isolated guest environments on top of a host OS. The main difference between the two technologies is that VMs fully emulate virtual hardware, and run a full-blown OS, whereas containers directly share the hardware and kernel with their guests. On the VM side, a hypervisor makes isolated virtualized slices of the hardware available. There are generally two types of hypervisors: ‘‘Type ’’ runs directly on the bare metal of the hardware, while ‘‘Type 2’’ runs as an additional layer of software within a standard OS [17].

Containers, in contrast, make available protected virtual partitions of the operating system. Two containers can run on the same operating system without knowing that they are sharing resources because each has separate abstracted networking layer, processes and so on [17]. Additionally, containers avoid the overhead due to virtualized hardware requested by hypervisor-based virtualization, thus enabling fast initialization and dense deployment of services, as experimented in [18]. To network applications, the consequence is that containers occupy much less resources and have lower virtualization overhead (i.e., latency, CPU cycles, memory waste, jitter) than VMs, leading to increased performance and deployment density in MEC scenarios. As a drawback, containers are much less adaptable, e.g., a Linux container cannot run on a Windows server [11], and applications relying on specific kernel features cannot be deployed to all hosts (even if the OS is of the same flavor). This limitation is a

serious drawback to MEC scenarios as the MEC hosts must be compatible with the guest, and the handover decision process must take this aspect into consideration.

III. SERVICE MIGRATION DIFFICULTIES

Service migration faces many difficulties. The fundamental problem facing VMs migration between hosts is the need to stop them, even if it is only for a short time. This might not have a big effect on services that are not accessed by users during that time. However, it significantly impacts running services on the VMs being moved during the live migration. Two performance parameters are crucial in this situation, namely downtime (i.e. the amount of time the VM will be stopped during the migration), and total migration time which is the time needed to complete the whole process of VM migration. This process encompasses the following steps:

1) Establish connectivity (e.g., layer 2 for intra-data-center operations) between the hosts.
2) Transfer the whole disk state.
3) Transfer the memory state of the VM to the target host as the source continues running without interruption.
4) Once the disk state and most of the memory states have been transferred, freeze the VM execution for the final transition of remaining memory dirty pages and processor states to the target server [20].

In some scenarios, where storage access is not unified or at least not federated, the entire disk image may need to be transferred and provisioned at the destination. This situation is highly sub-optimal and will present a serious performance limitation. For a standard Linux system, this rapidly implies a few more Gigabytes of data, with the respective delay imposed by the bandwidth available between the origin and destination. This imposes heavy traffic load to the network connection and results in significant service blackouts as analyzed in [19]. For service supporting high mobility, where frequent migration may be required, this approach is not advised. Live migration of memory is one of the key problems in LAN migration since in many well-designed scenarios the disk state of the VM is accessible (or transferable) at both the source and destination machines (e.g. via shared storage). Especially because, although storage can be centralized, memory is always local and must always move. The first stage of the migration is called iterative pre-copy, which iteratively copies the memory pages of the VM from the source to the destination. Since the application is still running uninterrupted during this stage, memory pages may get dirtied after they are copied. Hence, after each round, all newly dirtied pages are re-sent in the next round. The iterative process continues to the point where the residual dirty state is considered ''small''. At this point, the source VM is paused and the residual state is finally sent over. The destination VM can resume execution starting from where the source VM has paused [21].

In the Wide Area Network (WAN) case, the Internet Protocol (IP) address of the VM must change since the destination host is part of a different network subnet; maintaining network transparency is, therefore, more difficult. A similar problem exists in traditional IP-based networks due to user mobility, and the need to change user IP address. This is traditionally solved using triangle routing techniques such as in Mobile IP [22]. However, such approaches result in transferring packets through longer paths leading to additional delays. These mechanisms are not suitable for edge computing environments where the main motive for migration is reducing application latency. Additionally, when considering moving a VM across WANs, low bandwidth and high latency over network connections may drastically degrade the performance of the VM migration and consequently the QoS/QoE levels of applications [20].

IV. QOS AWARENESS IN THE VIRTUALIZED EDGE

The term QoS describes certain characteristics of the service provided to users such as availability, packet loss, bit rate, throughput, delay, reliability and jitter. Numerous factors influence the overall delay experienced by the end user. These are 1) transmission delay: is the time required for transmitting the task and results packets, and the time these packets spend traveling between the user and the edge server [23], 2) queuing scheduling delay which defines the time the tasks await in a buffer or queue before it is processed and it grows exponentially with resource utilization [25], 3) processing delay which is the task execution time at the MEC node [24] 4) propagation delay. In a three-layer cloud computing system, the overall delay is defined as the summation of computation and communication delays, where communication delay encompasses communication latency over the radio between the User Equipment (UE) and the edge node, and backhaul latency from the edge node to the cloud. Meanwhile, the computation delay is dependent on the queuing system used and is the summation of queuing time and task execution time [26]–[28]. Additionally, the term *service response time* identify the time elapsed since a user sends a task and he receives a result.

A principal contributor to service quality degradation in environments with virtualized edges is users' mobility. When the user moves away from the edge node providing the service, the communication delay will increase accordingly.

Many techniques have been proposed in literature with the aim at maintaining an optimal quality of service for mobile users in VE environments such as a) path selection with handovers [29], b) user-based inter-cloud service migration [30], and c) service migration [31].

The mechanism of Path Selection with Handovers (PSwH) was first proposed in [29]. This approach focuses on finding the optimal communication path between the mobile user and the cloud enhanced small-cell, also referred to as small cell eNodeB (SCeNBs or cloudlet). The authors distinguish the computing cell where the computation tasks are performed from the serving cell which is the attachment point with the user. The serving SCeNBs are the gateway nodes. Each gateway distributes the offloaded computation data to a number of the computing cells through the network infrastructure.

The serving cell is selected as the SCeNB with the highest RSSI. If the user moves during computation or if the radio channel and backhaul link load or quality change, the serving cell is updated. Thus, the proposed scheme forces the UE to perform handover if it leads to shortening of data transfer time and better quality of service [29].

On the contrary to this approach, where the management is the provider responsibility, the mechanism proposed in [30], referred to as User-Side Cloud Service Management (UCSM), is dependent on the user for management. The UCSM process is divided into two phases (pre-interaction and post-interaction phases). In the pre-interaction phase, the user selects a service from the available services and in the post-interaction phase, the user monitors the selected service by using the early warning framework to verify that the desired or promised QoS levels are maintained. If the user determines that the selected service fails to meet these QoS levels, it needs to take appropriate steps for service management to ensure the achievement of the desired outcomes. The UCSM framework consists of three modules, namely: (1) service monitoring; (2) QoS forecasting and early warning; and (3) decision-making.

Service migration approaches depend on moving cloud services between edge nodes so that the service can get closer to mobile subscribers, thus minimizing the communication delay. Follow-Me Cloud (FMC) [32] was one of the first approaches to exploit this notion to reduce latency experienced by mobile users. FMC enables migrating virtual instances, hosting services, across federated Data Centers (DCs) or edge nodes in unison with the user's movements, to enhance the quality of service offered to the user.

A combination of PSwH and service migration is proposed in [33], where the dynamic nature of the system is handled by first predicting the user's movement, and then exploiting this prediction for dynamic VM placement before the UE starts offloading. Thus, minimizing the delay since VM migration, if done while offloading is in progress, would increase the offloading delay. In [33], however, when the UE starts offloading its task, the VM will be already prepared at the suitable eNB. From the perspective of latency, the suitable eNB would be the eNB with the highest SINR, as the channel quality directly relates to the communication channel capacity. Then, the UE exploits the PSwH algorithm enhanced with mobility prediction, to select a suitable communication path (i.e., the serving eNB) in order to further reduce offloading delay. Cooperation is achieved by starting the algorithm for dynamic VM placement in-between offloading of two consecutive tasks, when certain radio conditions are met (SINR is below a given threshold). This takes advantage of both approaches (VM migration and PSwH).

It must be noticed that the offloading delay perceived by users is the sum of the communication delay (uplink and downlink), ingress/egress delay in the interfaces, and the computational delay. Ingress and egress delay are related to the specificities of the communication technology at a lower layer and are not specific to these scenarios. Most approaches that rely on the previously mentioned mechanisms (PSwH, UCSM, and VM migration) try to reduce latency by reducing the communication delay between the user and the VE. On the other hand, there are techniques that aim to minimize the overall delay by minimizing the computation delay such as, job scheduling [34], [35], latency-aware edge node selection [36]–[40], optimizing edge host placement [41], [42], and dynamic resource management [43].

In this paper we restrict our study on service migration approach in the virtualized edge environment, focusing on QoS-aware migrations, reviewing the methodologies proposed and discussing the challenges facing them. We then highlight issues and problems kept unresolved.

A. QOS-SPECIFIC SERVICE MIGRATION ISSUES

Traditional service migrations normally occurring between different data centers are usually triggered for purposes such as power consumption reduction [44]–[46], optimizing resources utilization [47]–[49], load balancing [50], or as a resilience mechanism [10]. In contrast, the VE environment stresses out the importance of QoS often considering delay a principal indicator. Therefore, most works aiming at sustaining QoS are delay-oriented.

Service degradation in VE is mostly a consequence of users' mobility where the increased distance between the service edge host and the mobile user will lead to more delay. Thus, instigating the orchestrator or edge manager to perform service instantiation in a location closer to the user. This notion was first proposed in Follow Me Edge (FME) [20], [51], where the service is instantiated in harmony with the user mobility, thus sustaining the QoS by ensuring a minimum distance between the service and the user. The fundamental problem with realizing this FME approach is taking the decision of whether to instantiate/migrate the service. Another additional hurdle is deciding the optimal timing and placement for the service replica and the total number of the replicas needed to be deployed. The uncertainty in user mobility, as well as possible variation of the migration and transmission costs, complicate taking such decision.

Another problem facing VM migration emerges from the distributed nature of the service infrastructure in the VE, where several edge data centers, that might be owned by different infrastructure providers, should be able to collaborate with each other. It is necessary to develop standards that specify how the different elements of the architecture can collaborate with each other, and also how the VMs can access certain information (e.g. context and host information) regardless of their deployment place [52].

Managing the VM life-cycle and resource allocation requirements in the edge node is also a critical issue. Due to users' mobility, optimal allocation of the computational resources at the base stations might change over time. QoS can be enhanced by provisioning more resources to the user, instead of migrating the whole service to another edge node. Finding the optimal placement, timing, and the number

of replicas can be decided by formulating an optimization problem depicting the trade-off between the benefits and costs of migration procedure [51]. There are already various works, in the areas of FC and MEC, that define and resolve optimization problems, whose goal is to improve the distribution of VMs over a set of edge computation hosts, minimizing a certain metric of resource utilization. These mechanisms also define when VMs need to be replicated, migrated or merged. The following subsection provides an overview of the solutions conducted up until now regarding QoS-aware service migration in VE.

B. QOS-AWARE SERVICE MIGRATION APPROACHES

As mentioned before, most works related to QoS-aware service migration in a VEs environment consider delay as the principal indicator of performance. Some works try to minimize the delay by adding QoS-awareness to the migration decision mechanism. Other works focus instead on the migration mechanism itself, trying to reduce the time needed to migrate the whole service. Following, we discuss both types of approaches.

1) DECISION-MAKING ORIENTED SOLUTIONS

Each VM hosting a service has a different migration cost in terms of resources allocation (such as CPU, memory, and bandwidth), and the duration needed to execute a full migration. This forms the main dilemma facing QoS-aware dynamic service placement/migration in a VE: establishing an optimal balance between the service migration cost and benefits. This is formulated and solved as an optimization problem using different techniques such as Markov Decision Problem (MDP), stochastic shortest path problems, dynamic programming or Lyapunov optimization techniques.

MDP, in particular, was used extensively to formulate the optimization problem for service migration in MEC. For example, [32] proposed an MDP-based model for service migration to implement the concept of Follow Me Cloud [51]. The proposed model allows the formulation of policies that achieve a good trade-off between user QoE and cost incurred by service migration. The main goal was to allow the network controller to optimally decide if a service needs to be migrated or not when the concerned UE is at a certain distance from the source Data Center. Numerical results show that the proposed service migration decision mechanism always achieves the maximum expected gain compared to two other policies, namely the policy that triggers service migration each time a UE enters into a new service area, and the policy that launches service migration only when the UE is on a specific distance from the source Data Center.

The problem with this approach is that it was only simulated considering one dimension (1-D) mobility model. Additionally, the solution is found with standard MDP solution approaches such as value and policy iteration. These standard solutions can become time-consuming when the number of states in the MDP is large [53].

Dynamic service instantiation to reduce delay for a mobile user was also suggested by other authors [54]. Distance-based MDP was used to model an optimization problem, in order to find the best placement for the new service instance. The authors establish a decoupling property of the MDP that reduces it to two separate MDPs on disjoint state spaces. Then, they design an online control algorithm for the decoupled problems using Lyapunov optimization techniques. The metrics taken into consideration are reconfiguration cost, transmission cost and back-end transmission cost, subject to meeting the average delay constraint. The transmission and reconfiguration costs are defined as a function of the distance (measured by the smallest number of hops) between different cells. The approach was tested against three alternative approaches that include never/always migrate policies and a myopic policy, that chooses actions to minimize the one-timeslot cost, and proved to outperform them. It was also tested using real-world mobility traces and synthetic traces obtained assuming random-walk user mobility.

The algorithm [54] does not require any statistical knowledge of the system parameters and can be implemented efficiently. This contrasts with traditional solution methods (such as dynamic programming) that require extensive statistical knowledge and are computationally prohibitive. The algorithm takes into consideration multiple edges, users, applications and demands on applications, comprising an interesting alternative to the traditional methods.

A threshold policy-based mechanism for service migration in mobile micro-clouds was proposed in [55]. It tries to optimally decide where to migrate the service when the mobile user has moved from one area to another in order to minimize latency. The problem was also modeled as a MDP. The authors proved that the threshold policy is the optimal policy for service migration when the mobile subscriber follows a one-dimensional asymmetric random walk mobility model. A threshold policy means the service is migrated to an edge more closely located to the user, when the distance between the service and user locations exceeds a threshold, and not migrate otherwise. The authors first proved the existence of an optimal threshold policy and then proposed an algorithm with polynomial time-complexity for finding the optimal thresholds.

The polynomial time-complexity used is independent of the discount factor. Thus, different from the standard algorithms for solving MDPs such as value iteration or policy iteration, in which the time-complexity is generally dependent on the discount factor, and they can only be shown to have polynomial time-complexity when the discount factor is regarded as a constant.

The problem with this solution [55] is that the authors neglected migration time, and only considered a 1-D mobility model. However, they extended their work in [56], [57], where a distance-based MDP was formulated, in order to design optimal service migration policies following the users' mobility in MEC environments. With this approach, the user

conforms to a uniform 2-D random walk mobility model on an infinite space. The cost function considered is restricted to migration cost and transmission cost, which is incurred by the user for connecting to the currently active service instance. That is related to the distance between the service and the user after possible migration. Further works [56], [57] were also tested against the never-migrate policy, the always migrates policy when the user and the service are at different locations, and the myopic policy. Real-world traces are used, and the results show that the proposed method brings discounted sum costs that are very close to the optimum. It also shows improvement in computation time compared to the policy iteration approaches.

The disadvantage of this enhancement [56] is that it only focuses on choosing if and where to migrate the service considering a single user accessing a single service. It is applicable to multiple users only if they run separate copies of the service. This disadvantage is further amplified because the authors assumed the time to perform migration is negligible when compared to the time-scale of node mobility. In reality, migration can take tens of seconds, while mobility can occur in seconds or milliseconds, especially when considering car or train travelers.

Additionally, distance-based MDP models, in general, do not fully support real-time mobile applications, due to them ruling out vital factors from the migration decision: 1) network state and 2) server state [59].

Furthermore, the MDP formulation requires that the user mobility has to follow (or at least can be approximated by) a Markov chain model and that the transition probabilities of this chain are known. Moreover, the methods for simplifying and approximating the MDP only work for a specific class of cost functions [53].

To overcome some of these drawbacks, the authors proposed an alternative approach [58], where they assumed that the cost is related to a finite set of parameters, which can include the user' locations and preferences, system load, database locations, among several other parameters. The focus was on the case where there is an underlying mechanism to predict the future values of these parameters. The anticipated future costs of each configuration can be found, in which each configuration identifies a specific placement sequence of service instances. The goal was to find the optimal placement (configuration) of instances to minimize the average cost over time.

In contrast to slot-based decisions in the class of MDP approaches, the solution [58] makes decisions on the basis of a look-ahead window, whose size is optimized as a function of the prediction accuracy. Within the look-ahead window, optimal placement decisions are made using dynamic programming based on the predicted future costs.

An offline algorithm, that solves for the optimal configuration in a specific look-ahead time-window, was first proposed. Then, an online approximation algorithm with polynomial time-complexity is presented to find the placement in real-time whenever an instance arrives. The effectiveness of the proposed approach was evaluated by simulations with both synthetic and real-world user-mobility traces. The method can work with more general cost functions, as long as future costs are predictable with known accuracy [53], [58]. However, these predictions are hard to be computed accurately in real environments, and this difficulty increases with network dynamics and the heterogeneity of MEC applications.

Another approach that relies on MDP is SEGUE [59], whose primary objective is to consistently meet reliable QoS standards for mobile users. It aims for optimal service migration in real time scenarios of dynamic environment. The optimization problem is formulated using MDP and incorporates both network and server states.

The system resolves service migration problems by using four individual processing modules. The first module is the State Collection module, which collects edge cloud network states and server workload. A reliable predicted QoS that accommodate a certain degree of mobile uncertainty is achieved through the QoS Prediction Module. The third module is the Edge Cloud Selection Module, which chooses the best edge cloud to migrate with using the aforementioned variables provided by the state collection and QoS prediction modules as inputs. Once the best edge cloud is selected, the Service Migration Module migrates the service from the source edge cloud to the optimal edge cloud. SEGUE performance was evaluated through an augmented reality application. The results demonstrate that SEGUE reduces the response time of this application significantly compared to the lowest load migration model and the least hop migration model.

Although this approach is more realistic than others by considering the server and the network situation, it suffers from a considerable computation overhead due to the prediction module which should predict the QoS to each application running on each user. Additionally, collecting the state for all mobile users raises the question about the scalability of such approach.

All previously mentioned approaches formulate the optimization problem using MDP. MDPs suffers from several drawbacks. First, it requires extensive knowledge of the statistics of the user mobility and request arrival processes that can be impractical to obtain in a dynamic network. Second, even with prior knowledge, the resulting problem can be computationally challenging to solve. Finally, any change in the statistics would make the previous solution suboptimal and require a new calculation to find the optimal solution [54].

A performance-aware service migration approach is presented in INDICES (INtelligent Deployment for ubIquitous Cloud and Edge Services) [61]. The aim is to move cloud-based interactive services hosted in the centralized Cloud Data Center (CDC) to a Micro Data Center (MDC). This approach highlights an interesting problem called performance interference, which is an indirect consequence of service migration. Performance interference is an inherent

property of any virtualized system that is caused by co-located applications contending for resources thereby leading to performance degradation [62]. The objective of INDICES is to assure the service level objectives for all the applications while minimizing the overall deployment cost. This cost includes the monetary allocation cost, and the operational costs of the running servers, such as the need for power and cooling. These objectives are met using an optimization problem, which is solved using a two-level cooperative and online process between system-level artifacts. INDICES is further extended in [43], to be a part of a whole edge cloud architecture.

Some approaches do not limit their focus on minimizing the communication delay by moving the service closer to the user, but also minimizing the computation delay. For example [23], [63] propose a technique for minimizing service delay in MEC by lowering the time for transmission and the time for processing simultaneously.

A mathematical model is formulated for evaluating service delay, which is expressed as the sum of processing delay and transmission delay. From this model, the factors affecting each type of delay are identified. It was found that the number of VM servers hosted by each VE node significantly affect the processing delay. Thus, VM migration is an effective technique to control the average processing delay. Additionally, it was shown that transmission delay is a function of the number of accommodated users and the transmission power of the VE nodes. Therefore, Transmission Power Control and VM migration (corresponding to user association) are candidates for managing transmission delay. Based on that, by finding the optimal values for these parameters, the service delay can be minimized. An optimization problem is formulated and solved by consecutively applying the partial derivatives method.

Through the optimal values of the parameters, the proposal is capable of controlling the transmission power of the VE nodes and deciding how many virtualized servers each of them can host. After, the management server proceeds to update user association and compares every association with its previous state, before the proposed procedure was executed. For all users that changed association, their corresponding virtualized servers are migrated from the original VE to the new one.

The main disadvantage of this approach is that the authors assume a VE environment where each VE node hosts a single VM for each accommodated user. This assumption is not the usual case of a real-world scenario. Additionally, the scenario is restricted with two VE nodes and has questionable ability to scale on more. User's mobility is not taken into consideration and the procedure proposed introduces overhead and its execution time must be configured wisely. Furthermore, this approach does not take into account the service downtime resulting from migrating the VM and consider it irrelevant compared to transmission and computing delay, although this must be considered as an important aspect of the computing delay.

In addition, as authors noted in [64], this approach is only feasible for small amounts of edge nodes since that number corresponds to the number of transmission power levels and consequently decision variables. Moreover, brute force has a complexity that is exponential on the number of decision variables. Linear Programming techniques cannot be applied considering this is not a linear equation system, and derivatives and integrals only work with few decision variables or it will become too complex. This lead the authors to extend this work later in [64] and use instead the Particle Swarm Optimization (PSO) model [65], an Artificial Intelligence technique, to achieve a low execution time and high efficiency. The approach was tested against No Migration approach and the Conventional approach and proved to outperform them.

Another approach that focuses on both transmission and computation delay is [38], where authors design an algorithm for selecting the computing cell that increases user's satisfaction with the experienced delay due to data transmission and computing. The algorithm selects the computing cell based on a combination of users' requirements, application parameters, and the edge host status. At the same time, the approach tries to achieve load balance among individual cells equally to minimize overhead. The proposed algorithm, denoted as Application Considering Algorithm (ACA), exploits both cluster state parameters obtained from all SCeNBces and also parameters of the offloaded task derived from the offloading requests.

PRIMAL PRofIt Maximization Avatar pLacement [66] proposes a cloudlet network architecture, with the aim at providing ubiquitous computing resources to UEs while maintaining low End-To-End (E2E) delay. Each UE subscribes to one Avatar, a high-performance VM hosted in the cloudlet. On the top of the cloudlets, a Software Defined Networks (SDN) based cellular core network has been proposed to provide the most efficient communications paths between the VMs hosted by the different cloudlets as well as between UEs in different Base Stations (BSs).

The migration decision-making algorithm aims at optimizing the trade-off between the migration gain and the migration cost. The gain is described as a reduction of the E2E delay between the UE's BS and the UE's cloudlet (in which the UE's Avatar or VM is located). Meanwhile, the migration cost is considered to be proportional to the total migration time and the cost coefficient. The cost coefficient is the weighted sum of utilization of different resources (bandwidth, memory, disk I/O and CPU).

The problem is found to be a quadratic assignment problem which is proven to be NP-hard and then reducible to the PRIMAL problem, which is a concave quadratic optimization. Authors use the Mixed-Integer Quadratic Programming (MIQP) tool in CPLEX solver to find the heuristic solution of PRIMAL.

PRIMAL achieves the highest profit when compared to the other two Avatar placement strategies, i.e., FAR (which tries to minimize the E2E delay by neglecting the migration cost) and Static (which tries to minimize the migration cost without

considering the E2E delay). However, the approach did not consider the cloudlet delay [67].

All previous approaches follow the reactive strategy of VM migration, while other approaches try to predict beforehand the user's movement and make a migration decision based on this prediction as in [68]. In this work, the authors propose an algorithm that dynamically places the VMs considering the load of eNodeBs (eNBs) and selects a communication path between the UE and the eNB with the allocated VM. The algorithm is based on MDP and exploits mobility prediction. For each eNB in close proximity to the user, the SINR is predicted by applying SINRmap [69] on predicted UE's mobility. Communication capacity is predicted from predicted SINR and the number of UEs utilizing communication resources.

For each VM placement decision required, the eNB with the highest available capacity is selected, then the eNB with the highest predicted gain in capacity is selected for VM placement. Following the selection of eNB for VM placement, VM migration delay is predicted, and the option with the lower delay between the instantiation of a new VM and VM migration is selected. The user is then forced to do a handover to the eNB providing better service. This method was tested against other already mentioned proposals [53] and [29] and proved to outperform them in terms of the average offloading delay. Other proactive techniques are proposed in [13], [70], [71].

To support ultra-short latency applications, authors of [13] consider proactive migration in a MEC cellular environment. The basic idea is to guarantee fast relocation by deploying multiple replicas of the user service in neighboring edge nodes, so to make services readily available if the handover to a different edge host is necessary. In this way, the long migration time required by on-demand service relocation could be drastically reduced. The mobility is predicted based on handover probability. An integer linear optimization problem was formulated and two solutions for replication-based service migration were proposed: the min-RM approach aims at minimizing the QoE degradation during user handover, and min-NSR approach favors the reduction of service replication cost. Simulation results proved the efficiency of each solution in achieving its design goal. However, the main problem with this approach is the restricted scenario where all the involved users require the same typology of service. Furthermore, only one service is associated with each user.

The authors extended their work by proposing a whole framework which implements this migration mechanism in [70] and [71]. The framework aims to support low-latency service provisioning by deploying container-based instances at the network edge.

Instead of migrating the whole service as suggested in [13], the framework relies on maintaining replicas of specific services to reduce the overall service migration in case of user mobility. Then the services instances are activated proactively after anticipating the user's mobility. It also provides synchronization of the data among different replicas in case of storage-dependent applications.

This framework has some challenges, such as an appropriate analytical model to optimize the number of service replicas has not been defined. Efficient placement of the replicas at different edge nodes was not discussed. It also does not provide the support for stateful applications.

A proactive migration of virtualized functions in response to device mobility is proposed in [72]. The approach uses a predictive hand-off mechanism that moves a specific virtualized function into the most suitable MEC node before the user request. The suitability of the edge node is based on the consideration of multiple aspects, such as network statistics, availability, and recoverability. By dynamic proactive hand-off scheme, the service is prepared in advance in the next MEC node. This minimizes the down time due to virtualized function migration. The predictive hand-off is complemented by a reactive behavior, necessary for the few cases when mobile devices request the migration and the needed VMs has not been migrated in advance, e.g, in the case of unexpected users' movements: service hand-off is performed in this case via requesting the discovery of a nearby MEC node.

To enable proactive migration, users location data (latitude and longitude) is stored on the global cloud to compose temporal data series. Periodically, the cloud performs data analysis on those historical time series, in order to predict next users' locations. To this purpose, a polynomial non-linear regression technique is used. The choice of using a regression-oriented technique instead of other discrete statistical models, e.g., a Markov model, is to have a more accurate prediction, depending on historical data and implicitly considering other elements such as direction/speed of movement, temporal/spatial data locality, and the delta between consecutive sensed locations. However, regression analysis requires more resources than discrete models, both for storing large amounts of data and processing them.

A placement and migration method for providers of infrastructures that incorporate cloud and fog resources was presented in MigCEP [73]. It ensures application-defined End-to-End latency limitations and reduces network utilization by arranging the migration ahead of time. The plan is triggered when the latency restrictions are no longer preserved on the current broker (edge device where the VM is placed). A time graph is created, where the vertexes are the edge devices which can host the VM, the directed edges are weighted with the costs that are expected to occur in a time interval (the average bandwidth-delay products weighted with that interval). This comprises the expected migration costs and expected placement costs at the broker. Vertexes are omitted when the associated broker is not taken into consideration to host the VM. This is the case when latency restrictions are not expected to be preserved, or when local information on the migration targets resource reservations indicates that not enough resources are available to execute the operator at the potential target. The shortest path

is determined. The migration itself starts out by sending the VM and immutable state to the selected migration target. The plan can be generated either locally at each operator or at a central coordinator, e.g., at a dedicated node in the cloud. A locally generated plan imposes an additional overhead related to a periodic exchange of plans (and feedback) with neighbors. The plan depends on a learned connection pattern that describes how probable it is for a source or consumer to change its connection to another leaf broker, after staying connected for a certain amount of time to its current leaf broker. Such learned probabilities are maintained in a graph, where each vertex represents a broker and has an edge to all neighboring brokers the source or consumer can connect to next. Each edge in the graph is annotated with the probability to change the connection and the average time a source or consumer is connected to a broker before it connects to the neighbor.

The approach is tested against two approaches: i) the static approach that does not perform any migration, and ii) the greedy approach that greedily select every few seconds the broker with the best placement cost. The results show that MigCEP outperforms both of them.

The main problem with this work is the lack of optimization due to the need for creating a time graph and executing the algorithm periodically. It also has a significant signaling overhead when the plan is exchanged between neighboring nodes. Further optimization is needed.

A different solution for VM migration [74] has the overall goal of minimizing service initiation time, which is the time from when the Mobile Device (MD) discovers a cloudlet to the time the MD starts offloading data. Four main steps are involved for the service initiation time: bind the MD with the cloudlet, transfer the VM overlay, decompress the VM overlay, and apply the VM overlay to the base VM to launch the VM. In order to reduce the delay caused by the service initiation time following the user movement away from the source cloudlet, the authors propose a seamless live VM migration technique between neighboring cloudlets. This VM migration is achieved with the prior knowledge of the migrating VM IP address in the destination cloudlet and with the use of Multipath Transport Protocol (MPTCP).

It was shown that the RTT behavior changes during migration. Thus, it is possible to distinguish: 1) Initial RTT, which is the average RTT between the migrating VM and the mobile device through the WAN before the migration process is started. 2) Actual average RTT, which is the average RTT during the migration process, and 3) Delta RTT, which is the difference between the actual average RTT and the initial RTT. The algorithm for migration decision-making considers three main parameters; the service initiation time, the VM migration time and the RTT during VM migration. If the RTT measured during the migration process is less than 150 msec, it means that the user is having a good QoE even through the WAN. Regardless of the total estimated VM migration time, the VM migration process will be performed. There is no need to trigger VM migration if the actual RTT during VM migration is greater than 1s since the service becomes useless for the user.

The algorithm must be implemented in each cloudlet. The estimation of the total VM migration time and delta RTT can be achieved through profiling. Applications and network resources profiler should be deployed in the system, which registers all VM migration results. Over time, the profiler provides precise information. Each cloudlet also can share their profiles to learn a wide variety of situations within a short time.

A number of experiments using emulation have demonstrated that the proposed approach [74] can be realized to support actual live VM migration. Further, the work presented an investigation of the performance of VM migration, including total live migration time, VM downtime, throughput, RTT and the performance of the TCP protocol.

TABLE 1. Comparing QoS-aware service migration approaches focus and types.

Approach	Focus	Migration Type
[32]	Transmission delay	Reactive
[54]	Transmission delay	Reactive
[55]	Transmission delay	Reactive
[56]	Transmission delay	Reactive
[58]	Queuing and propagation delays	Offline and real-time placement
[59]	E2E and response time	Proactive
[61]	Execution duration,Server performance	Proactive
[63]	Transmission Delay,Computation delay	Reactive
[64]	Transmission Delay, Computation delay	Reactive
[33]	Offloading delay (communication and computing delay)	Proactive
[13]	Service downtime	Proactive
[72]	Service downtime	Proactive and Reactive
[73]	Transmission and Computation delay	Proactive
[74]	Service down time	Reactive
[66]	E2E delay	Reactive

In Tables 1, we compare the different QoS-aware service migration approaches previously mentioned in terms of the focus and type. In Table 2, we compare the different methodologies followed and their performances. Furthermore, Table 3 summarizes the main features of the scenarios considered when evaluating such approaches.

2) MIGRATION STRATEGY ORIENTED SOLUTIONS

Implementing an appropriate strategy to perform the migration is of substantial importance, and can offer many benefits for the overall migration process optimization. Therefore, some research was focused solely on migration technique such as [75], which proposed a three-layer framework for migrating services in MEC. The goal was to reduce the service downtime and overall migration time. This is done by distinguishing the service application layer from a base layer, which includes the guest OS kernel and all other components that are required to make the service application self-contained. The running state of an application is placed in a third layer called the instance layer. Then a copy of the base layer is kept in every MEC node so it does not have to be migrated in every migration. A base package with no services installed can have a large size (hundreds of megabytes and a few gigabytes for LXC and KVM). By avoiding migrating this layer, the service downtime is drastically minimized. The principal benefit of this framework is that it is built based on functionalities readily available in most container and VM

TABLE 2. Comparing QoS-aware service migration approaches methodologies and performance.

Approach	Method	Mobility model	Costs	Outperforms
[32]	Distance-Based MDP	1D	unidentified	Always migrate, Threshold-based migration
[54]	Distance-based MDP, Lyapunov optimization	Random walk, Real world traces	Transmission costs, Reconfiguration cost	Never migrate, Always migrate, Myopic policy
[55]	MDP	1D random walk	Distance-related Migration cost, Transmission cost	Never migrate, Always migrate
[56]	Distance-Based MDP	2D random walk, real-world traces	Distance related Migration cost, Transmission cost	Never migrate, Always migrate, Myopic policy
[58]	Dynamic programming	Synthetic instance-arrival/departure, Real-world traces	local and migration resource consumptions	Never migrate, Always migrate
[59]	MDP	1D Real mobility	Service down time, Server workload, Link BW	Lowest load migration, Least hop migration
[61]	Linear programming	–	Monetary cost, Operational cost	Least hope, Minimum load server-selection approaches
[63]	Partial derivatives method	Static scenario	–	No migration Workload balancing VM migration
[64]	Particle Swarm Optimization	Static scenario	–	No migration Workload balancing VM migration
[33]	MDP	Random mobility	Cloudlet state, application requirement	[30], [60], [73],
[13]	Integer Linear Optimization	–	QoE degradation, Deployment cost	–
[72]	Non-linear regression	-	Network statistics, Availability	–
[73]	Shortest path	Realistic traffic	Migration cost, Placement cost	No migration, Greedy approach
[66]	Mixed-Integer Quadratic Programming	Random waypoint	Migration cost, Resources utilization	FAR, static

TABLE 3. Comparing the scenarios considered for testing migration approaches.

Approach	Management Entity	Scenario Features
[32]	FMC controller	Single user, Single destination
[54]	System-wide controller	Multi-users, Multi-services
[55]	Central entity	Single user, Single service
[56]	MEC controller	Single user, Single service,
[58]	System controller	Multi-users, One service per user
[59]	Edge Cloud Selection Module	Multi-users
[61]	Local managers and the global manager	Single user, Single service,
[63]	A management server,	Multiple applications, Single VM per user
[64]	Central office	Multiple applications, Single VM per user
[33]	Unidentified	Multi-users
[13]	Unidentified	Multi-users, One service per user, Same type of service
[72]	Global cloud for proactive Server Manager for reactive	Single service, Single user
[73]	Locally at each operator or, At a dedicated node in the cloud	Multi-users
[74]	Every Cloudlet	Emulation experiments
[66]	SDN controller	Multiple users,One vm for user

technologies that are popular today, so that one does not need to have extensive knowledge on the technical details of the container or VM implementation to apply this framework. A further work [11] provides more a comprehensive discussion and experimentation results of this mechanism. However, keeping a copy of the base layer in each MEC node requires prior knowledge of the required OSs and kernels needed for the services. In addition, this approach suffers

from drops of resource utilization due to the large amount of data pre-installed in MEC nodes which may not be used.

Another approach is proposed in [76], which exploits Multipath Transport Control Protocol (MPTCP) [77] to speed up migration process and overcome both the network transparency and bandwidth limitations of VM migrations in edge clouds. The approach utilizes the aggregated throughput provided by MPTCP to parallelize migration, thus reducing migration time. It also relies on MPTCP resilience properties to lower latency after migration by automatically switching active network connections to the VM's new address, thus removing the need for a persistent tunnel. The system introduces a new iterative pre-copy algorithm for disk and memory, optimized for the lower bandwidths of edge clouds. The prototype performance was demonstrated on both a distributed public cloud and a lab-based edge cloud. Experiments show that it reduces migration time by up to 50% and in some scenarios increases clients post-migration throughput by almost 6x compared to typical tunneling approaches.

MPTCP is also used in other work [79], which adopts an approach involving container-based cloudlets, and uses CRIU (Checkpoint/Restore in Userspace) as the chosen tool for migrating containers. CRIU is a tool that can be used to suspend an ongoing application (or part of it) and checkpoint it to a hard drive as a collection of files. Then, these files can be restored, enabling the application to run from the point it was suspended at. Additionally, to guarantee a better success of container migration, this paper leverages MPTCP. With its help, multiple paths, i.e. subflows between the container (the source) and the remote host (the destination) can be established. If one subflow goes down during migration, the traffic would seamlessly switch to other subflows, ensuring the continuity of the established MPTCP connection. However, the results of the experimental work on this approach showed very big migration time (more than 240s), which is intolerable for latency-sensitive applications.

In [80], a seamless live VM migration is achieved with the prior knowledge of the migrating VM IP address in the destination edge host and with MPTCP. Reference [80] proposes two additional features on top of MPTCP. The first feature is to have two virtual interfaces instead of one on each VM. One interface operates during normal operation and the other one operates when a VM migration is triggered. The second added feature is to let the VM instance knows its future IP address. The new VM IP address is the next odd IP address of the paired mobile device IP address. Such IP address assignment mechanisms help in reducing TCP disruption and improve overall performance. However, the IP assignment technique can be problematic in scenarios with larger scales. For example, assuming one VM has the IP address 10.0.0.3 and another with the IP address 10.0.0.4, and both are required to be migrated at the same time. Taking the next odd IP address for each VM as a future IP address will create a conflict in this scenario.

Some approaches try to reduce the total VM migration time by using techniques inspired by VM synthesis. Dynamic VM synthesis [5] leverages the fact that most VM images are derived from a small number of widely-used base VM images (typically Linux or Windows) that can pre-deployed on the edge hosts. The desired application-specific VM image, referred to as the "launch VM", is initially created through a slow offline process that involves the installation of an application-specific software into a base VM. This also includes the installation of the dynamically-linked libraries and toolchains. The VM overlay is the binary difference between a launch VM and its base VM. The VM overlay's small size is the key to the previous work on rapid provisioning [82]. When instantiating the VM, the launch VM is reconstituted by applying the overlay to its respective base VM. Synthesizing a VM faces many challenges such as security, trust and compatibility problems as discussed in [5]. In addition to it raising the question of what is the adequate edge node sizing should be.

A real-time VM handoff is proposed in [78] as a technique for seamlessly transferring VM-encapsulated execution to a more optimal offload site, to cope with the consequences of user mobility. This approach is inspired by VM synthesis. However, rather than overlay creation being a one-time offline operation, a series of overlays are generated afresh at runtime on the source edge node during the course of a single VM handoff. The time for overlay creation, which was ignored in earlier work because it was an offline operation, now becomes a significant limiting factor. In addition, the tuning parameters (such as compression algorithm) used in overlay creation are dynamically re-optimized at runtime in order to reflect the current relative costs of network transmission and edge node computation. Thus, although VM handoff have similar concepts to VM synthesis, it represents a substantial new mechanism on its own.

The VM handoff solution [78] accounts for three considerations:(a) optimizing for total handoff time rather than downtime; (b) dynamically adapting to WAN bandwidth and edge host load, and (c) leveraging existing VM state at edge node. To efficiently discover and encode the differences between current VM state at the source and already-present VM state at the destination, a pipeline of processing stages is used. This delta encoding is then de-duplicated and compressed (using parallelization wherever possible) and then transferred. The algorithms and parameters used in these stages are chosen to match current processing resources and network bandwidth.

The study is made under the assumption that the all-important variables of when and where to migrate were known. These assumptions cannot be made in the real world since the conditions that can trigger the migration of a virtualized service may vary widely. The tradeoff between the cost of migration and any real QoS improvement must be taken into consideration. Additionally, despite the remarkable efforts, the results still show significant values of total migration times, which could cause a notable degradation of QoE in MEC environment, since degraded E2E latency persists until the end of the operation [70]. Furthermore, the approach is reactive, i.e., the migration is triggered only when the

TABLE 4. Migration approaches focusing on the migration technique.

Reference	Technique	Advantages	Disadvantages
Machen [75]	VM synthesis	Reduced migration time	Inefficient resource utilization
Chaufournier [76]	MPTCP, Pre-copy of disk and memory	-Reduced migration time by up to 50% -Increased after-migration throughput	Inefficient resource utilization
Qiu [79]	MPTCP	-Improved migration resilience -Reduced migration time	Evaluated on containers inside VMs not physical hosts
Teka [80]	MPTCP VM IP assignment	Reduce down time	A problematic IP assignment technique
Satyanarayanan [5]	Dynamic VM synthesis	Improved migration process	Security, trust and compatibility issues
Ha [78]	Modified VM synthesis	Improves total migration time	Still high total migration time
Bellavista [72]	VM synthesis	Improved QoE	Inefficient resource utilization
Sun [81]	Pre-deployed VM replica	Minimizing E2E delay	Inefficient resource utilization

user moves to a different edge node. This means that some degradation in the service may already be noted before the migration is triggered.

The same concept of VM synthesis was exploited in another work [72]. In this work, the experiments determined that the overall time to complete handoff has to be 172.52 seconds, with a total VM downtime (generating mobile service unavailability and service interruption) perceived by the client of 1.60 seconds, because of the proactive migration and VM synthesis described previously. While these values look promising, they still required improvements from at least two orders of magnitude in order to support scenarios with real-time mobility.

Another technique [81] was proposed where virtual disk migration is avoided during the VM/avatar handoff process. This is done by pre-deploying a number of the VMs replicas among edge nodes. The edge node which contains one of the VM replicas is defined as the Avatar's available cloudlet. Thus, an Avatar can only be migrated to its available cloudlets.

After deploying the replicas of each UE's Avatar among cloudlets, the Avatar can be handed off among its available cloudlets based on the UE's location. Optimally, when the UE roams into a new location, the Avatar will be handed off to the cloudlet, which incurs the lowest E2E among all available nodes. However, each cloudlet has its CPU and memory capacity, and so the Avatar may not be handed off to the optimal one because the optimal cloudlet may not have sufficient residual capacity to host the Avatar. Therefore, an Avatar replica placement problem is formulated and the LatEncy Aware Replica placemeNt (LEARN) algorithm is proposed to solve the problem. By considering the capacity limitation of each VE, another algorithm, named LatEncy aware Avatar hanDoff (LEAD), is proposed to optimally place UEs' Avatars among the cloudlets in each time slot such that the average E2E delay between UEs and Avatars is minimized.

Simulation results demonstrate that applying the LEARN algorithm in the cloudlet network architecture can significantly reduce the average E2E delay between UEs and their Avatars during the day as compared to the traditional centralized cloud architecture (i.e., all the UEs' Avatars are located in the central cloud). Furthermore, the LEAD algorithm can still maintain a low average E2E delay by selecting suitable parameters in terms of the number of the replicas for each Avatar and the capacity of each VE. We summarize the various techniques used for migration with the aim of reducing the migration downtime in Table 4.

V. TOOLS TO SUPPORT THE DESIGN OF NEW SOLUTIONS

Several simulators and tools have been introduced to facilitate the development of cloud computing and virtualization models such as CloudSim [83], GreenCloud [84], DCsim [85], iCanCloud [86], CloudNet [87], EMUSIM [88] and Eucalyptus [89]. On the other hand, simulation of edge computing is supported by a smaller number of tools such as EdgeCloudSim [90] and iFogSim [91]. In both cloud computing and edge computing, only a few simulators support scenarios involving VM migration.

While CloudSim is considered to be the most common platform for cloud computing simulation, it lacks a proper model for VM live migration. The VM in CloudSim is represented as an entity with a specific `id`, image size (MB), memory, bandwidth, MIPS, and set of CPUs. When the migration is triggered (in CloudSim) the migration delay is only considered to be as $delay = \frac{RAM}{\frac{BW}{2}}$, where half of the bandwidth (`BW`) is for transmitting the RAM, while the other half is for standard VM communication, without taking into consideration aspects such as the dirty page rate. Therefore, CloudSim is supporting only cold and not live migration.

GreenCloud [84] also implements VM migration in a simplified way, without the support of live migration. Here, the total size of the VM is divided into packets transferred from the source host to the target host, ignoring memory changes in the case the VM is still running, thus only considering idle VMs for migration.

VMs in DCsim [85] are also considered entities with specific storage, memory, CPU and bandwidth utilization. The migration process is only represented as an event between the source and the target, where the resources are allocated in the target and released from the source. DCsim [85] also considers a static service time penalty and CPU overhead as a result of the VM migration. These values are configurable and not calculated based on a mathematical model for each VM. iCanCloud [86], on the other hand, takes into consideration the rate of dirty pages of the VM, but does not take into

TABLE 5. Simulators with virtualization technology support.

Simulator	Extension of	Language	Cold Migration	Live Migration
CloudSim	Independent	Java	Yes	No
GreenCloud	NS-2	C++/TCL	Yes	No
DCSim	Independent	Java	Yes	No
iCanCloud	OMnet++	C++/Ned	Yes	Yes
CloudNet	Independent	Java	Yes	Yes
EdgeCloudSim	CloudSim	Java	No	No
iFogSim	CloudSim	Java	No	No
myIFogSim	iFogSim	Java	Yes	Yes

consideration the physical layer between the users and the cloud, and only represents the users as entities to submit jobs to VMs located in the cloud.

The same issue appears in CloudNet [87] where the dirty page rate is considered when estimating the total size of the VM to be migrated. However, the physical medium and its effect on the migration process are ignored, and the migration process is only considered as an event of allocation and deallocation of resources.

EdgeCloudSim [90] and iFogSim [91] are extensions to CloudSim that support edge and fog computing. Still, they have no support for VM or task migration between edge hosts. myiFogSim [92] is an extension to iFogSim to support virtual machine migration policies for mobile users. This software supports three VM migration techniques: 1) full VM migration where an image of the entire VM is created and delivered to the transport layer, 2) container migration, and 3) live migration based on live-migration post-copy, where the system and the user data that are not in the main memory are delivered first and then the data in the main memory is delivered [93]. The main problem with myiFogSim lies in its restriction to fog environments. Table 5 summarizes the characteristics of the previously mentioned tools.

Cloud Management Platforms (CMPs) have been provided by public cloud vendors aiming at managing the resources provided by the Infrastructure as a Service (IaaS) cloud. These platforms hold responsibility for monitoring, provisioning, orchestrating, and automating services and applications residing in the cloud.

IBM Cloud Orchestrator [94], AppFormix [95], and ServiceNow Cloud Management [96] are some of the common CMPs. There are also some open-source platforms such as Apache CloudStack [97], OpenStack [98], Cloudify [99] and OpenNebula [100]. These platforms provide utilities and solutions to support service migration as part of their functionality. In particular, OpenStack is very relevant to these scenarios, as it is being used as the underlying solution for most of the proof of concepts and integration work. Furthermore, all of the major virtualization providers offer independent live migration tools, such as VMware vMotion [101], Microsoft Hyper-V [102], Citrix XenMotion [103], and CloudEndure Migration [104], which could be used to implement dynamic solutions in virtualized edges.

VI. CHALLENGES IN SERVICE MIGRATION FOR VE IN 5G NETWORKS

Based on the careful analysis of the state of the art, we can identify some research challenges left unhandled or even unmentioned, which we find relevant to discuss. We consider that without solutions for these challenges, the reliability and confidence of VE enabled networks will be sub-par with traditional Telco networks. The consequence is that the milestones for 5G networks, especially those related to latency and reliability, will not be met in a near future. These challenges can be summarized as per the following paragraphs.

A. OPTIMAL IDENTIFICATION OF THE SERVICE TO BE MIGRATED

The problem of service migration requires the answers of three questions: when to migrate the service? where to migrate it? and which service should be migrated?. While the first two questions have been handled in several studies, the later has not been considered yet. Most approaches previously presented require the same service typology for multiple instances or restrict the scenario to a single service instance as can be noted from Table 3. However, in real-world scenarios, situations with multi-users running multiple services simultaneously are more common, or even the norm. These services may differ in criticality, delay restrictions, and resource allocation. This adds complexity to the decision-making process of the service migration.

B. DECISION-MAKING RESPONSIBILITY

Choosing the location of the entity responsible for deciding the service migration can be critical to the performance of migration decision making and execution.

Three options can be considered:

a) A localized approach i.e either the source or the destination edge device is responsible for triggering the migration. The source edge can trigger the migration in case of proactive migration. Meanwhile, the destination edge can trigger the migration after receiving information from the source edge or from the user.

b) A centralized approach of decision making. In this approach, the data center or controller entity is responsible for making the deciding of migration.

c) The last approach is the distributed approach, where intermediate entities residing at the edge of the network or one tier above the edge are responsible for monitoring edge devices and users equipment in certain areas and migrating the services when needed.

C. DECISION-MAKING ALGORITHM EXECUTION

Most service migration models recalculate the optimal edge cloud for mobile users randomly or periodically. The interval period for such recalculation has not been not fully analyzed or studied.

Nevertheless, determining the optimal recalculation interval is essential. Since running optimization models such as MDP is a computationally intensive task, short recalculation intervals introduce heavy overheads to the server. Conversely, longer recalculation intervals may translate into lazy migration resulting in periods of transgression of QoS guarantees.

Furthermore, proactive migration in VE environment requires real-time feeds about network and servers states. Collecting this information should also be performed at specific intervals. If such interval was not chosen wisely, it might result in flooding the network with overheads. This issue has not been discussed in any of the previous models.

D. STATEFUL SERVICE MIGRATION

Although this challenge has been mentioned before, it has not been studied thoroughly. Most works focused on stateless applications. However, from our experience, most applications are stateful in the sense that users will login, or have some specific preferences that customize service delivery. Bindings to databases, filesystems and to other services may also be considered. This makes the VM handover process more complex as not only the communication flows with the UE must be kept, but also the VM binding must be migrated. In some sense, the VM acts as a mobile entity, similar to a UE, doing handovers in the network edge.

E. MULTIPLE VM LIVE MIGRATIONS

Chain VM migrations and Network functions migration have not been studied thoroughly in the literature. In particular, it seems clear that VM migration for multiple VMs should not be done simultaneously, or at least should be done with care. Instantiation of a new VM will consume more resources than the one needed when the VM is in a steady state. Therefore, VM migration will have an impact on other VMs in the destination VE host, and this impact must be considered.

F. PROACTIVE/REACTIVE REPLICATION

Most approaches studied were focused on making migration-decision reactively. These approaches suffer from significant computation overhead and questionable scalability. On the other hand, the approaches that adopted the proactive approach, despite being a proper approach for providing minimum delay, suffer from a significant waste of resources. This is due to instantiating multiple replicas in different locations to handle mobility uncertainty. Most of these replicas might not be needed. This can be compensated if more sophisticated mobility prediction mechanisms are added.

G. POST-MIGRATION PROCESS

After taking the decision of migration, two additional procedures should be handled. They are a) handling IP address change (in case of WAN migration) and b) redirecting the traffic without any loss. Only few works have explicitly considered these procedures such as [80]. Taking an efficient way to assign the IP address can help optimize the migration procedure significantly.

H. COST DILEMMA

Most approaches assume specific structures of the cost function that are related to the locations of users and service instances. Additionally, most of the costs considered are presumed to be similar in all edges. However, the heterogeneity of the edge devices may cause these costs to differ extremely. This will make such cost functions incapable of presenting the actual cost, resulting in unoptimized decision.

I. MIGRATION, HAND-OFF AND OFFLOADING

All three concepts are highly depending on each other. Service migration is the process of transferring or migrating an edge node application to another node. The offloading describes the process where a mobile user can access and offload computation jobs to nearby edge host to be executed instead of executing them locally on the device. Offloading improves the performance and reduce local execution cost. The handoff is the process of transferring the communication link between the user and the network from one base station to another.

The common scenario for reactive migration is where a user is first connected to a source edge, offloading some of its tasks while on the move. The tasks are computed in a VM located on this edge host. The user's movement will result in a drop of Received Signal Strength Indicator (RSSI) perceived by the user and the possibility of getting closer to another edge with better RSSI. This will urge the user to demand a handoff process so it can be connected to this new edge (destination). However, this handoff will result into a longer path between the VM performing the computing and the user, where the computation results must transverse from the computing edge (source) to the destination edge to the user. To minimize the delay resulting from the longer path, VM migration is triggered.

To optimally handle the user movement in VE environments and to provide the least delay possible, all three aspects (offloading, handoff and migration) must be optimized simultaneously. However, even considering this to be vital for VE deployments, no complete framework has been proposed.

For future works this relationship should be highlighted, so to minimize the overall delay introduced from all three procedures. Additionally, more sophisticated mobility prediction mechanism should be utilized in the proactive service migration approaches.

J. SIMULATION PLATFORMS AND SUPPORT TOOLS

As discussed in section V, most cloud simulators provide a very basic implementation for VM migration. They consider a VM of a fixed size and disregard the memory and disk state changes. This represents cold migration processes only. Very few simulators provide an implementation to live VM migration, and even so, only in a very simple model.

VII. FUTURE WORK DIRECTIONS

Some of the identified challenges can benefit from existing mechanisms. For instance, handover prediction methods [105]–[107] can avail proactive service migration in choosing the target edge node to host the migrated service. Additionally, service downtime can be minimized by adopting novel migration mechanisms such as thread-based or paralleled live migrations [108], [109], which efficiently migrate memory intensive VMs by creating multiple threads parallel to the VM threads in the hypervisors at both source and destination hosts, coupled with compression techniques. Theses threads function as controllers, memory page transmitters and dirty pages transmitters and receivers. However, the hypervisors and in particular, the orchestrators of virtualized telco environments, are slow to treat migration and overall provisioning, requiring work to be developed to streamline interfaces and policies triggering migration.

VM migration time can be minimized by adopting several approaches such as a) leveraging the existing VM snapshots in backup servers [110], b) taking advantage of Storage Area Network (SAN) and transferring the attachment to the persistent data instead of the data itself, or even exploiting the existence of a Network Attached Storage (NAS) in simpler scenarios [111]. The development of intelligent solutions allowing multiple attachments to the same persisted object, with locking capabilities at a large scale, is still a challenge. Distributed file systems which can provide high performance over scenarios of heterogeneous coupling (high in the core, loose near the edge), are a strong research direction for future work.

Avoiding redundant re-transmission of memory pages by combining the migration process with dirty pages predictions was explored in several works [112]–[114], as well as memory compression techniques [115]–[117]. Still, as the infrastructure of a 5G network encompasses a large area (such as a country), deduplication exploring data locality from neighbor virtualization hosts can be explored here. This requires strong coordination but can provide rapid reconstruction of a migrated service if it is popular. In this context, the initial placement of edge servers can significantly influence service performance. The development of strategies for intelligent server placement and service deployment mechanisms is important, as it can help reduce access delay and may result in avoiding service migration altogether [37], [118], [119].

Joint optimization of computation offloading and service migration can minimize service downtime. For instance, executing computation tasks locally on the device, during the service freeze phase of the migration process, can help diminishing some of the service downtime. However, this requires tight integration of UE and the virtualized edge, with code offloading to be done in a bidirectional way. The works related to generalized offloading of code (and not functions), are still in their early stages, and clear solutions are required.

VIII. CONCLUSIONS

The research community is actively developing solutions capable of enabling the 5G vision, to which we require a flexible and low latency communication infrastructure. Network Virtualization techniques, especially when applied to the edge of the network are very promising.

We discussed a variety of works that focus on the added challenges brought by that vision. In particular, handover processes due to UE mobility will become extremely more expensive, and VM migration is a key aspect to be addressed. However, there are other aspects that must be catered for, such as the heuristics driving the decision processes, and the overall coordination of the process. We show that, while some solutions present opportunities for further developments, the area is still very far from a generalized solution, or even from a solution capable of providing overall mobility and service continuity in VE environments. This is particularly critical when considering strict QoS and QoE requirements, which will be increasingly dominant in future wireless cellular environments. We have identified some of the areas where research should be developed in order to overcome current technology shortcomings.

REFERENCES

[1] M. Fallgren *et al.*, *Scenarios, Requirements and KPIs for 5G Mobile and Wireless System*, document ICT-317669-METIS/D1.1, Apr. 2013.

[2] *Industry Proposal for a Public Private Partnership (PPP) in Horizon 2020 (Draft Version 2.1), Horizon 2020 Advanced 5G Network Infrastructure for the Future Internet PPP.Online*. Accessed: Mar. 25, 2019. [Online]. Available: http://www.networks-etp-eu/fileadmin/user_upload/Home/draft-PPP-proposal.pdf

[3] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[6] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *Proc. IEEE 9th Int. Conf. Mobile Ad-Hoc Sensor Netw.*, Dec. 2013, pp. 373–377.

[7] J. Liu, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Performance evaluation of integrated multi-access edge computing and fiber-wireless access networks," *IEEE Access*, vol. 6, pp. 30269–30279, 2018.

[8] H. Routaib, E. Badidi, M. Elmachkour, E. Sabir, and M. ElKoutbi, "Modeling and evaluating a cloudlet-based architecture for mobile cloud computing," in *Proc. 9th Int. Conf. Intell. Syst., Theories Appl. (SITA-14)*, May 2014, pp. 1–7.

[9] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics WebSyst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[10] A. Fischer, A. Fessi, G. Carle, and H. de Meer, "Wide-area virtual machine migration as resilience mechanism," in *Proc. IEEE 30th Symp. Reliable Distrib. Syst. Workshops*, Oct. 2011, pp. 72–77.

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

[12] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement. (NSDI)*, Berkeley, CA, USA, vol. 2, May 2005, pp. 273–286.

[13] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[14] V. Medina and J. M. Garcia, "A survey of migration mechanisms of virtual machines," *ACM Comput. Surv.*, vol. 46, no. 3, p. 30, 2014.

[15] A. Shribman and B. Hudzia, "Pre-copy and post-copy VM live migration for memory intensive applications," in *Proc. Euro-Par Parallel Process. Workshops*, in Lecture Notes in Computer Science. New York, NY, USA: Springer, 2012, pp. 539–547.

[16] C. Jo, E. Gustafsson, J. Son, and B. Egger, "Efficient live migration of virtual machines using shared storage," in *Proc. 9th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, Houston, TX, USA, 2013, pp. 41–50.

[17] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, 2014.

[18] R. Morabito, J. Kjallman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Mar. 2015, pp. 386–393.

[19] A. Ruprecht *et al.*, "Vm live migration at scale," in *Proc. 14th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, Mar. 2018, vol. 53, no. 3, pp. 45–56.

[20] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi, "Clouds of virtual machines in edge networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 63–70, Jul. 2013.

[21] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast transparent virtual machine migration in distributed edge clouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, 2017, pp. 1–13.

[22] C. Perkins, *IP Mobility Support for IPv4*, document 3344, 2002.

[23] T. G. Rodrigues *et al.*, "Towards a low-delay edge cloud computing through a combined communication and computation approach," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2016, pp. 1–5.

[24] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 3, Nov./Dec. 2004, pp. 1629–1634.

[25] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proc. 9th Int. Conf. Auton. Comput.*, 2012, pp. 145–154.

[26] Y. Hao *et al.* (2019). "Smart-edge-cocaco: Ai-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT." [Online]. Available: https://arxiv.org/abs/1901.02126

[27] G. Li, J. Wang, J. Wu, and J. Song, "Data processing delay optimization in mobile edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, Feb. 2018, Art. no. 6897523.

[28] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[29] Z. Becvar, J. Plachy, and P. Mach, "Path selection using handover in mobile networks with cloud-enabled small cells," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2014, pp. 1480–1485.

[30] Z. U. Rehman, O. K. Hussain, E. Chang, and T. Dillon, "Decision-making framework for user-based inter-cloud service migration," *Electron. Commerce Res. Appl.*, vol. 14, no. 6, pp. 523–531, 2015.

[31] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[32] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1350–1354.

[33] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.

[34] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.

[35] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[36] A. Mukherjee, D. De, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *IEEE Trans. Cloud Comput.*, vol. 7, no. 1, pp. 141–154, Jan./Mar. 2019.

[37] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.

[38] M. Vondra and Z. Becvar, "QoS-ensuring distribution of computation load among cloud-enabled small cells," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 197–203.

[39] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 1281–1290.

[40] M. Kwon, Z. Dou, W. Heinzelman, T. Soyata, H. Ba, and J. Shi, "Use of network latency profiling and redundancy for cloud server selection," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2014, pp. 826–832.

[41] L. Ma, J. Wu, and L. Chen, "DOTA: Delay bounded optimal Cloudlet deployment and user association in WMANs," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 196–203.

[42] E. Wong, S. Mondal, and G. Das, "Latency-aware optimisation framework for cloudlet placement," in *Proc. Int. Conf. Transparent Opt. Netw.*, Jul. 2017, pp. 1–2.

[43] S. Shekhar and A. Gokhale, "Dynamic resource management across cloud-edge resources for performance-sensitive applications," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 707–710.

[44] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud, Grid Comput. (CCGrid)*, May 2013, pp. 671–678.

[45] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, "Jettison: Efficient idle desktop consolidation with partial VM migration," in *Proc. 7th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2012, pp. 211–224.

[46] H. Shirayanagi, H. Yamada, and K. Kono, "Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2012, pp. 460–467.

[47] H. W. Choi, H. Kwak, A. Sohn, and K. Chung, "Autonomous learning for efficient resource utilization of dynamic VM migration," in *Proc. 22nd Annu. Int. Conf. Supercomput. (ICS)*, 2008, pp. 185–194.

[48] R. U. Ighare and R. Thool, "Threshold based dynamic resource allocation using virtual machine migration," *Int. J. Current Eng. Technol.*, vol. 5, no. 4, pp. 2603–2608, 2015.

[49] S. Seth and N. Singh, "Dynamic threshold-based dynamic resource allocation using multiple vm migration for cloud computing systems," in *Proc. Int. Conf. Inf., Commun. Comput. Technol.* Singapore: Springer, 2017, pp. 106–116.

[50] C.-T. Fan, W.-J. Wang, and Y.-S. Chang, "Agent-based service migration framework in hybrid cloud," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, Sep. 2011, pp. 887–892.

[51] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep./Oct. 2013.

[52] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[53] S. Wang, K. Chan, R. Urgaonkar, T. He, and K. K. Leung, "Emulation-based study of dynamic service placement in mobile micro-clouds," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Dec. 2015, pp. 1046–1051.

[54] R. Urgaonkar *et al.*, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015.

[55] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2014, pp. 835–840.

[56] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. 14th IFIP Netw. Conf.*, May 2015, pp. 1–9.

[57] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung. *Dynamic Service Migration in Mobile Edge-Clouds Based on Markov Decision Process*. Accessed: Mar. 25, 2019. [Online]. Available: https://sites.psu.edu/nsrg/files/2017/08/MigrationWithExponentialCost_TMCsub-14x6190.pdf

[58] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[59] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proc. Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2016, pp. 344–351.

[60] W. T. Tsai, P. Zhong, J. Elston, X. Bai, and Y. Chen, "Service replication strategies with mapreduce in clouds," in *Proc. 10th Int. Symp. Auto. Decentralized Syst. (ISADS)*, Mar. 2011, pp. 381–388.

[61] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, "INDICES: Exploiting edge resources for performance-aware cloud-hosted services," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 75–80.

[62] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 77–88, 2013.

[63] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[64] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "A PSO model with VM migration and transmission power control for low service delay in the multiple cloudlets ECC scenario," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[65] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, Mar. 2016.

[66] X. Sun and N. Ansari, "PRIMAL: Profit maximization avatar placement for mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[67] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.

[68] K. Wang, M. Shen, J. Cho, A. Banerjee, J. van der Merwe, and K. Webb, "MobiScud: A fast moving personal cloud in the mobile network," in *Proc. ACM SIGCOM 5th Workshop Things Cell. Oper. Appl. Challenges*, 2015, pp. 19–24.

[69] H. Li, S. Habibi, and G. Ascheid, "Handover prediction for long-term window scheduling based on SINR maps," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2013, pp. 917–921.

[70] I. Farris, T. Taleb, H. Flinck, and A. Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, 2017, Art. no. e3169.

[71] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[72] P. Bellavista, A. Zanni, and M. Solimando, "A migration-enhanced edge computing support for mobile devices in hostile environments," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 957–962.

[73] B. Ottenwalder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "MigCEP: Operator migration for mobility driven distributed complex event processing," in *Proc. 7th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS)*, 2013, pp. 183–194.

[74] F. Teka, C. H. Lung, and S. A. Ajila, "Nearby live virtual machine migration using Cloudlets and multipath TCP," *J. Cloud Comput.*, vol. 5, no. 1, p. 12, 2016.

[75] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Migrating running applications across mobile edge clouds: Poster," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2016, pp. 435–436.

[76] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast transparent virtual machine migration in distributed edge clouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, 2017, pp. 1–13.

[77] S. E. Barr, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *Proc. Int. Conf. Res. Netw.*, 2011, pp. 444–457.

[78] K. Ha *et al.*, "Adaptive VM handoff across cloudlets," School Comput. Sci., CMU, Pittsburgh, PA, USA, Tech. Rep. CMU-C S-15-113, 2015.

[79] Y. Qiu, C. H. Lung, S. Ajila, and P. Srivastava, "LXC container migration in cloudlets under multipath TCP," in *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2017, pp. 31–36.

[80] F. Teka, C.-H. Lung, and S. Ajila, "Seamless live virtual machine migration with cloudlets and multipath TCP," in *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 607–616.

[81] X. Sun and N. Ansari, "Avaptive avatar handoff in the cloudlet network," *IEEE Trans. Cloud Comput.*, to be published.

[82] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. MobSys*, Taipei, Taiwan, Jun. 2013, pp. 153–166.

[83] *CloudSim Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: http://www.cloudbus.org/cloudsim/

[84] *GreenCloud Simulator*. Accessed: Mar. 25, 2019 [Online]. Available: https://greencloud.gforge.uni.lu/install.html

[85] *DcSim Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/digs-uwo/dcsim

[86] *iCanCloud Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: http://icancloud.org/

[87] *CloudNet Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/dmitrygrig/CloudNet

[88] *EmuSim Framework*. Accessed: Mar. 25, 2019. [Online]. Available: http://www.cloudbus.org/cloudsim/emusim/

[89] *Eucalyptus Cloud Platform*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/eucalyptus/eucalyptus

[90] *EdgeCloudSim Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/CagataySonmez/EdgeCloudSim

[91] *iFogSim Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/Cloudslab/iFogSim

[92] *MyIFogSim Simulator*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/marciocomp/myifogsim

[93] M. M. Lopes, W. A. Higashino, M. A. M. Capretz, and L. F. Bittencourt, "MyiFogSim: A simulator for virtual machine migration in fog computing," in *Proc. 10th Int. Conf. Utility Cloud Comput.*, 2017, pp. 47–52.

[94] *IBM Cloud Orchestrator*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.ibm.com/us-en/marketplace/deployment-automationIBM-Orchestrator

[95] *AppFormix*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.juniper.net/uk/en/products-services/application-management-orchestration/appformix/

[96] *ServiceNow Cloud Management*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.servicenow.com/products/cloud-management.html

[97] *Apache CloudStack*. Accessed: Mar. 25, 2019. [Online]. Available: https://cloudstack.apache.org/

[98] *OpenStack*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.openstack.org/

[99] *Cloudify*. Accessed: Mar. 25, 2019. [Online]. Available: https://cloudify.co/

[100] *OpenNebula*. Accessed: Mar. 25, 2019. [Online]. Available: https://opennebula.org/

[101] *VMware vMotion*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.vmware.com/latam/products/vsphere/vmotion.html

[102] *Microsoft Hyper-V*. Accessed: Mar. 25, 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview

[103] *Citrix XenMotion*. Accessed: Mar. 25, 2019. [Online]. Available: https://wiki.xenproject.org/wiki/MigrationXenMotion

[104] *CloudEndure Migration*. Accessed: Mar. 25, 2019. [Online]. Available: https://www.cloudendure.com/live-migration/CloudEndure

[105] N. AlJeri and A. Boukerche, "An efficient movement-based handover prediction scheme for hierarchical mobile IPv6 in VANETs," in *Proc. 15th ACM Int. Symp. Perform. Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw.*, 2018, pp. 47–54.

[106] Z. Becvar, "Efficiency of handover prediction based on handover history," *J. Converg. Inf. Technol.*, vol. 4, no. 4, pp. 41–47, 2009.

[107] X. Chen, F. Mériaux, and S. Valentin, "Predicting a user's next cell with supervised learning based on channel states," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2013, pp. 36–40.

[108] K. Chanchio and P. Thaenkaew, "Time-bound, thread-based live migration of virtual machines," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, May 2014, pp. 364–373.
[109] C. Kim, C. Jeon, W. Lee, and S. Yang, "A parallel migration scheme for fast virtual machine relocation on a cloud cluster," *J. Super Comput.*, vol. 71, no. 12, pp. 4623–4645, 2015.
[110] Y. Yang, B. Mao, H. Jiang, Y. Yang, H. Luo, and S. Wu, "SnapMig: Accelerating VM live storage migration by leveraging the existing VM snapshots in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1416–1427, Jun. 2018.
[111] P. Thakre and V. Sahare, "Optimization of migration time affected storage overheads during VM live migration using network attached storage device," *Int. J. Eng. Comput. Sci.*, vol. 6, no. 4, pp. 20806–20808, 2017.
[112] G.-F. Sun, J.-H. Gu, J.-H. Hu, and T.-H. Zhao, "Improvement of live memory migration mechanism for virtual machine based on pre-copy," *Comput. Eng.*, vol. 37, no. 13, pp. 254–265, 2011.
[113] E. P. Zaw and N. L. Thein, "Improved live VM migration using LRU and splay tree algorithm," *Int. J. Comput. Sci. Telecommun.*, vol. 3, no. 3, pp. 1–7, 2012.
[114] Y. Cui, Y. Lin, Y. Guo, R. Li, and Z. Wang, "Optimizing live migration of virtual machines with context based prediction algorithm," in *Proc. 1st Int. Workshop Cloud Comput. Inf. Secur.*, in Advances in Intelligent Systems Research, 2013, pp. 1–4.
[115] H. Jin, L. Deng, S. Wu, X. Shi, H. Chen, and X. Pan, "MECOM: Live migration of virtual machines by adaptively compressing memory pages," *Future Gener. Comput. Syst.*, vol. 38, pp. 23–25, Sep. 2014.
[116] L. Hu, J. Zhao, G. Xu, Y. Ding, and J. Chu, "HMDC: Live virtual machine migration based on hybrid memory copy and delta compression," *Appl. Math. Inf. Sci.*, vol. 7, no. 2L, pp. 639–646, 2013.
[117] M. R. Desai and H. B. Patel, "Efficient virtual machine migration in cloud computing," in *Proc. 5th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Apr. 2015, pp. 1015–1019.
[118] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.
[119] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, May 2019.

HADEEL ABDAH received the bachelor's degree in telecommunication engineering from Tishreen University, Syria, and the master's degree in computer networks and telematic services from the University of Minho, Portugal. She is currently pursuing the Ph.D. degree with the University of Aveiro, Portugal. She is a former Lecturer with the Institute of Computer Technology and the Intermediate Institute of Computer Engineering of Latakia. She was a Research Collaborator with Centro Algoritmi, from 2015 to 2016, for two years. She is currently with the Instituto de Telecomunicacoes of Aveiro, as a Team Member of CRUISE Project. Her research interests include wireless sensor networks, software-defined networks, network functions virtualization, and mobile computing.

JOÃO PAULO BARRACA received the bachelor's degree in computers and telematics, the master's degree in electronics and telecommunications, and the Ph.D. degree in informatics engineering, in 2012. He is currently an Assistant Professor with the University of Aveiro, where he has been a Lecturer, since 2008. At the same time, he has conducted research with the Telecommunications Institute, and led the TN-AV, in 2015 and 2016, for two years. He has close to 100 publications, including both peer reviewed and technical reports. His current research interest includes security and protocols for the Internet of Things in cloud environments, with a focus in solutions for software-defined networks in future networking environments. He has participated in more than 20 projects (eight EU funded), either developing novel concepts, or applying these concepts in innovative products and solutions, involving the design of the IoT platforms for telecom operators and water monitoring networks. He leads the FCT/CAPES DEVNF Project that addressed novel 5G orchestration in Portugal, the local teams in EU LIFE-PAYT, participates in EU AENEAS (European science cloud for astronomy), the local team in the P2020 at the CRUISE Project, the Security Team at the P2020-SOCIAL, participates in the EU Interreg CISMOB, focusing in smart transportation and connected buses, the Engage SKA Research Infrastructure, and the Square Kilometer Array System Team, having lead activities for the instrument cloud platform, among a dozen of other innovation projects.

RUI L. AGUIAR received the bachelor's degree in telecommunication engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, in 1990 and 2001, respectively.

He was previously an Adjunct Professor with INI, Carnegie Mellon University. He was a Visiting Research Scholar with Universidade Federal de Uberlandia, Brazil, for three years. He is currently a Full Professor with the University of Aveiro, responsible for the networking area. He is coordinating a research line nationwide with the Instituto de Telecomunicacoes, in the areas of networks and multimedia. He is leading the Technological Platform on Connected Communities, a regional cross-disciplinary industry-oriented activity on smart environments. His current research interest includes the implementation of advanced wireless networks and systems, with a special emphasis on 5G networks and the future Internet. He has more than 450 published papers in his research areas, including standardization contributions to the IEEE and IETF. He has served as Technical and General Chair for several conferences, including the IEEE, ACM, and IFIP. He is regularly invited for keynotes on 5G and future Internet networks. He sits on the TPC of all major IEEE ComSoc conferences. He has extensive participation in national and international projects, of which the best example is his position as the Chief Architect of the IST Daidalos Project, and he has extensive participation in industry technology transfer actions. He is currently associated with the 5G PPP Infrastructure Association. He is the Current Chair of the Steering Board, Networld2020 ETP. He is a Senior Member of Portugal ComSoc Chapter Chair, and a member of the ACM. He is an Associate Editor of Wiley's ETT and *Wireless Networks* (Springer), and he has helped on the launch of Elsevier's *ICT Express*. He is a Chartered Engineer, and he has acted as a Consultant of several major operators, as a Technology Advisor to bootstrap several SMEs, and as an expert to several public bodies, both on the societal and on the judiciary branches. He currently sits on the Advisory Board of several EU-projects and research units.