ARPA: AN AUTONOMOUS RENDERER PLACEMENT ALGORITHM IN DISTRIBUTED MULTIMEDIA FOG NETWORKS WITH DELAY GUARANTEES

Athanasios Tsipis[1] and Konstantinos Oikonomou[2]
[1]Dept. of Digital Media and Communication, Ionian University, Kefalonia, Greece, atsipis@ionio.gr, [2]Dept. of Informatics, Ionian University, Corfu, Greece, okon@ionio.gr

NOTE: Corresponding author: Athanasios Tsipis, atsipis@ionio.gr

Abstract – *Multimedia cloud computing has emerged as a popular paradigm for the support of delay-intolerable immersive multimedia applications with high-end three-dimensional rendering. To that end, fog computing offers distributed computational offloading solutions, by positioning rendering servers in close proximity to end users promising in this way continuous service provision, that is otherwise not easily attainable under the strictly centralized cloud-only model. Yet, in order to alleviate the multimedia providers from unnecessary capital expenditure, a strategic placement approach of the servers at the fog layer must be implemented, that can effectively cope both with the network dynamics and the overall imposed deployment cost, and still adhere to the delay bounds set forth by the multimedia application. In this paper, we formally formulate the problem as a facility location problem using constrained optimization over a finite time horizon. We then theoretically analyze the minimum acceptable conditions necessary for a decentralized location of the servers, utilizing solely local information around their immediate neighborhood, that iteratively leads to better solutions. Based on the analysis, we propose a distributed algorithm, namely the Autonomous Renderer Placement Algorithm (ARPA), to address it. ARPA employs localized service relocation to shift the placement according to simple rules that designate elastic migration, replication, and complementary consolidation of the underlying renderers. Simulation results under diversified deployment scenarios, as well as trace-driven comparisons against other approaches, testify to ARPA's accountability in obeying the delay limits and fast converge in finite time slots to a placement solution that both outperforms the baseline alternatives and is close to the optimal one, rendering it suitable for scaling up and down to meet the current demands of the offered multimedia applications.*

Keywords – Delay guarantees, distributed facility location, fog computing networks, multimedia cloud computing, rendering server placement, scalability

1. INTRODUCTION

The advent of *Cloud Computing* (CC) has taken a huge leap towards the direction of offering end users hardware independence and providing software services on-the-fly [1]. As one of its most fast-evolving paradigms nowadays, *Multimedia Cloud Computing* [2] has turned the tables when it comes to the provision of high-quality graphics multimedia services, such as immersive cloud gaming [3] and Virtual/Augmented/Mixed Reality (xR) applications [4]. Fueled by elastic resource provision, it provides affordable, flexible and intuitive ways for accessing multimedia services anywhere and anytime, usually through advanced virtualization of service procedures that allow interactive multimedia applications to be rendered in the cloud and the produced three-dimensional (3D) frames to be subsequently streamed as video sequences back to the users' clients for display [5].

Despite its enormous benefits, cloud-based multimedia remains prone to network delays that significantly impact the overall Quality of Service (QoS) of the users [6]. Many works strive to tackle this challenge via various smart and autonomous resource location-allocation methodologies, such as constrained optimization [7], fairness-based utilization [8], game category adaptability [9], and deep reinforcement learning [10].

Intuitively, for any cloud multimedia model to be branded as successful, it requires an ultra-low delay and cost-efficient design [11]. However, this is not easily attainable under conventional strictly centralized architectures, especially when considering heterogeneous network environments [12]. Hence, alternative approaches turn to the potential offered by *Fog Computing* (FC), or its complementary *(Mobile-) Edge Computing* (EC) [13], by positioning geographically distributed rendering servers [14], hereinafter also named *fog renderers*, in close proximity to the users' clients, leading to hybrid models for the support of *cloud/fog multimedia* services [15].

Previous work, e.g., [16, 17, 18], has indicated a clear tendency in the sense that fog renderers can indeed achieve significant delay reduction. Nevertheless, a significant driver for the adoption of the approach relates to the capital expenditures that are required by the multimedia providers for hosting such services in the FC layer. Thus, in order to truly actuate their potentiality, an intelligent placement, that optimally locates the fog renderers and is resilient to dynamic network fluctuations, must be considered during deployment while keeping in mind the necessity for constant QoS guarantees in terms of network delay, a factor that is crucial for delivering high-end interactive multimedia applications.

Still, the discovery of an optimal fog rendering server placement with the goal of minimizing *provision cost* for multimedia clients and *deployment cost* for multimedia providers, while maintaining an acceptable degree of QoS in terms of delay across all fog nodes and, therefore, users, is complex. Generally speaking, optimization problems like those previously mentioned are $\mathcal{NP}$-hard for general graphs (even when the service cost is considered to be the Euclidean distance from a node to the nearest service host [19]), requiring global knowledge of the network topology and generated demand workload. In fact, they can be straightforwardly reduced to well-studied *Facility Location Problems* (FLP) [20]. Given that cloud/fog multimedia systems are inherently extremely dynamic [14], then centralized solutions, where a super-entity with global information solves the problem in one iteration, are deemed unscalable and unsuitable for real-world, large-scale multimedia applications due to the exceedingly high cost for the continuous recomputation of the optimal solutions when changes do occur.

That being so, in the current work we focus on solving the aforementioned FLP in a distributed manner, based solely on *local network information* available to each fog renderer, regarding the rendering demands forwarded by fog nodes in their immediate neighborhood (one-hop away). We assume that such local information is trivial to acquire since it can be easily collected utilizing standard monitoring tools (e.g., Skitter), DNS queries, routing tables, or even be shared by the nodes using multicast, a control message exchange mechanism, or other communication protocols with minimal overhead [21]. By doing so, different from centralized solutions, we allow each rendering service to *relocate* towards its optimal location in a few time slots based solely on localized topology knowledge, by solving per time slot a small-scale FLP instance across its neighborhood considering the demand flows and perceived delays of the fog nodes within.

Our contributions are both on the theoretical and practical front. From a theoretical point of view, we initially formulate the problem as a constrained Uncapacitated Facility Location (UFL) problem over a finite time horizon, which for convenience hereafter we suitably refer to as the "Latency Bounded Uncapacitated Fog Renderer Location" (LB-UFRL) problem. We then mathematically analyze the worst-case conditions under which, during the aftermath of an initial arbitrary placement, a rendering service relocation (either being *migration*, *replication* or *consolidation*), that is based explicitly on localized knowledge, poses a positive impact on total cost (i.e., the sum of service provision and deployment cost) reduction after its occurrence. We also provide a clear methodology that ensures during the end of the placement process no violation regarding our latency bounds will manifest. Based on the analysis provided, we go a step further by proposing a novel distributed algorithm to address the LB-UFRL, namely the "Autonomous Renderer Placement Algorithm" (ARPA). The ARPA's relocation rules, time complexity, convergence and approximation properties are also thoroughly discussed. From a practical point of view, through comprehensive simulations with synthetic topologies, we offer both microscopic and macroscopic insights relating to ARPA's location adaptation under diverse fog network scenarios. The algorithm is demonstrated to be capable of fast convergence and accuracy, closely approximating the optimal solution in a scalable manner (even when changes to the network conditions do occur), and always abiding by the given constraints. To further increase validity, we also present trace-driven evaluations with real-world topology data, where we extensively compare ARPA against several existing alternatives and prove its superiority in producing near-optimal placements to the LB-UFRL, that are comparable only to the ones produced by the purely centralized greedy approach which employs exhaustive searches.

The remainder of the paper is organized as follows. Section 2 provides the necessary literature background on service location, while Section 3 outlines the considered system architecture for the realization of cloud/fog multimedia immersive applications. Section 4 includes the fog network's description and definitions, whereas Section 5 models the LB-UFRL placement problem as a constrained UFL. Section 6 offers the theoretical location analysis, and Section 7 sheds light on the proposed ARPA that capitalizes on the analytical findings. Section 8 showcases the conducted simulations and results, and Section 9 encloses the evaluative trace-driven comparisons with past approaches. Finally, Section 10 concludes the paper and draws the line for future research. All lemma and theorem proofs are found in the supplementary Appendix A.

2. RELATED WORK

The turn towards cloudification and virtualization of resources and services has sprung numerous works that are the subject of frontier research in the direction of service placement in the general ecosystem of cloud-enabled Content Delivery Networks (CDNs) [22]. Most of these focus purely on the cloud side, attempting to optimally locate Virtual Machines (VMs), responsible for running the services, at the most favorable locations in the cloud data centers that optimize several conditions, such as energy consumption (e.g., [23]), bandwidth utilization (e.g., [24]), traffic load (e.g., [25]), reliability (e.g., [26]), etc. For a complete survey, the reader may consult the recent work in [27], where many works driven by specific use preferences depending on the application environment are enlisted. However, with the emergence of Fog and Edge, or even the Internet of Things (IoT), the focus is gradually shifted toward the outskirts of the network. Thus, new research manifests that addresses the problem at a lower level, near the end users [28].

Such problems are typically tackled as an instance of well-studied FLPs [20], which are characterized by high complexity ($\mathcal{NP}$-hard for general graphs). Despite the ongoing efforts, centralized approaches, such as greedy algorithms [29, 30], are not always applicable to the ex-

tremely diversified and heterogeneous modern networks, as they impose a prohibitive communication overhead associated with collecting global knowledge regarding the topology and service demands. The difficulty is amplified when considering special cases regarding resource provision or response time specifications [31], which increase complexity with the procurement of additional upper (or lower) thresholds regarding service availability. As such, many attempts, like the one presented here, focus nowadays on distributed approaches [32], that take advantage of local information, to offer greater scalability during service placement, e.g., [33, 34, 35], rendering them flexible and adaptable to the frequent changes.

With that said, multimedia cloud computing [2], being one of the most fast-evolving cloud paradigms nowadays (either in the sense of cloud gaming or xR application streaming), has caught the eye of the research community giving birth to an increasing volume of works relating to its opportunities and challenges. A fundamental issue, however, relates to the location of the service responsible for 3D video rendering or next-generation content streaming, which are considered as computationally intensive processes [36]. Consequently, it comes with no surprise that placement issues (especially with the uprising of FC/EC and the wide proliferation of mobile smart devices), focusing on *computational offloading* and subsequently placement are at the core of this research (e.g., [15, 16]), usually in the form of discovering optimal locations for placing VMs, data resources, rendering servers, or application services that in sequence will maximize QoS for users and profits for multimedia vendors. Some of these works are listed next.

2.1 Past placement approaches

The authors in [37] present a series of VM placement models, that target the maximization of capital profits for the service providers in contrast with the maximization of Quality of Experience (QoE) for users. Their experiments showed that the first target is easier to achieve under public cloud systems, whereas the second is more suitable for closed ones. On a similar note, the authors in [12] propose an improved gray wolf algorithm to solve the VM provisioning problem for multiplayer games in geographically distributed data centers, in a way that minimizes interaction delay among users and the electrical expenses for cloud service providers. Both relate to this work since the proposed ARPA also attempts to find a balanced trade-off between the two contradicting goals, i.e., the clients' QoS optimization (with provision cost reduction) and the providers' profit maximization (with deployment cost reduction).

Understandably, QoE plays a significant role in immersive media streaming applications [38]. A typical example that addresses it is found in [7] where, similar to the present work, the authors formulate the cost-optimal placement of multimedia content distribution services using constrained optimization. On a different approach, the authors in [39] attempt to maximize QoE by minimizing gaming impairments through judicious migration of edge services that ensures bandwidth and migration costs abide by the available capacities and budgets. Likewise, this work also considers the pair-wise render cost reduction between fog nodes and their assigned fog renderers, which in turn leads to total provision minimization, as explained later. Simultaneously, there exists work that considers the combination of fog/edge and IoT. Characteristic examples are the recent works of [40, 41] where the authors, similarly to the present work, optimize QoS impairments based on pre-obtained service demands before each new iteration at the EC and FC layers respectively.

The studies of [14] and [42] explore the cost-efficient server allocation problem in cloud gaming, employing heuristics and approximation algorithms to overcome in the former barriers imposed by the servers' rental and bandwidth costs while in the latter the storage and software costs, respectively. It is noteworthy that in both of these works, a concluding remark is that placing servers on random, on the closest, or on the cheapest locations is far from optimal. On the contrary, it is proven that a combinatorial analysis of all must be explored, acknowledging the timely accessibility limitations of the servers. Current work addresses the above by autonomously and proactively dispensing the renderers in a fog-assisted cloud multimedia system, abiding by stringent latency constraints. When considering the FC (which is the layer of interest in current work although ARPA is easily extendable to EC) or EC landscape, the close-to-ours recent study in [43] proposes a bio-inspired genetic algorithm approach to locate fog game servers in a way that minimizes their number, respecting the available capacity and resource usage. On the other hand, in [44], the authors deal with the xR group engagement services and explore their dynamic placement with graph-theoretic approaches in Mobile Edge Cloudlets, offering constant performance guarantees considering discrete-time prediction windows. Similarly, in [45] the authors consider the allocation of microservices in the fog environment to tackle issues impacting the experience of xR clients and offer quality assurance. Towards QoE augmentation, the work of [46] additionally investigates different features of IoT devices to prioritize different application placement requests according to user expectations and then calculate the capabilities of fog instances considering their current status for efficient content delivery. The preceding research finds relevance to current work in the sense that ARPA also attempts to optimize QoS for all users connected to the fog nodes, respecting in the process multimedia-oriented delay expectations that will lead to increased QoE. Recently, in [47], the focus shifted on energy consumption and service availability regarding the active fog servers, and hence a placement and migration technique was highlighted capable of dynamically adapting to user mobility. Energy efficiency was also the goal in [48] and [49]. In

the former, two distributed approaches were provided that rely on Markov approximation for the placement of fog microservices. Neighboring fog nodes can then autonomously decide upon microservice movement or swapping until a state is reached where energy consumption and communication costs are minimized. In the latter, the authors devised a particle swarm optimization algorithm to maximize energy profits under constraints relating to the capacity and proximity of the edge servers. The authors in [50] employ a lightweight framework and propose two greedy fog placement algorithms for the dynamic deployment of applications, that generate low costs subject to different QoS restrictions. Like ARPA, this particular framework takes into account the aggregate demands of the underlying IoT nodes to modify the placement periodically.

As mentioned, cloud multimedia performance is also tightly associated with fluctuations in workload among the participating entities. Therefore, load balancing is another crucial objective that is regularly encountered. For instance, in [51] a location-priority algorithm to address the capacitated FLP version of the edge server allocation problem is proposed, while in [52] the goal is to maximize service accessibility with simultaneous deployment minimization, by merging redundant applications, alleviating in this way edge servers from unnecessary computation. Although we do not explicitly consider capacities in the present work, nevertheless, ARPA's placement properties, as discussed later, are heavily influenced by the workload witnessed in the renderers' neighborhood in terms of service demand accumulation. Thus, ARPA can be easily extended to incorporate such dependencies.

Taking a different research direction, the recently proposed approach found in [53], showcases the potentiality offered by Artificial Intelligence (AI). The authors utilize an AI-enabling mechanism for the placement of multimedia service instances in mobile edge computing ecosystems, that considers user mobility for path prediction and then employs a meta-heuristic binary particle swarm optimization approach to achieve a trade-off between QoE and overall network deployment cost. In [54] the k-FLP is tackled for edge provisioning of interactive services such as gaming applications, wherein the authors propose a greedy centralized approach that aims at maximizing the percentage of covered users, by iteratively selecting sites that satisfy the most mobile users. In the recent work of [55], a placement solution is followed based on the intuition that hub nodes which are characterized by high Betweenness Centrality (BC) and obey some delay metrics regarding the provision of next-generation services (like xR) are suitable candidates for deploying edge servers. Similarly, in [56] the authors consider the cost-effective edge server placement in wireless Metropolitan Area Networks (MANs) and propose a novel approach based on edge division into clusters, by solving a dominating set problem in graph theory under QoS requirements. Our work finds relevance to this previous work. In fact, we also adopt elements from graph theory to enable fast convergence in a finite number of time slots which, as already stated, also ensures compliance with latency bounds that impact multimedia rendering performance, by employing elastic relocation rules, i.e., migration, replication, and consolidation.

Such relocation mechanics, based on localized information, have been the goal of past literature, e.g., [57, 58, 59, 21]. To the best of the authors' knowledge, the closest work to this paper is the one in [60], which has shown to yield reduced overall costs (even if at the expense of not always providing optimal solutions) for general cloud computing networks, while being resilient to their high dynamicity. The presented algorithm, thus, relates to these research approaches but greatly diversifies itself from them in order to capture the idiosyncrasies of cloud/fog multimedia systems and offer agile rendering service placement, that can fast adapt to network oscillations with constant latency guarantees during the termination of the location process.

2.2 Literature comparison

To summarize the previous studies, we now provide a taxonomy on the basis of their application environment. Moreover, we showcase additional classifiers regarding their location mechanics (if any), as well as optimization objectives and constraints.

Table 1 enlists the literature on service placement and highlights the contributing elements of our approach. For the different classifiers, we note:

- *Environment*: Indicates the targeted landscape, either cloud, edge (including mobile edge), fog, or other (e.g., web, CDNs, data grid, etc.).
- *Decentralized*: Indicates whether the approach is distributed or centralized.
- *Dynamic*: Indicates whether the approach is dynamic and scalable to network changes or user mobility.
- *Migration*: Shows if the approach allows migration, exchange, hand-off, movement, or shift of services.
- *Placement Adaptation*: Due to terminology diversity amongst the different studies, it is further divided into two subclasses as (i) *Scale-up*, if the approach facilitates an expansion phase, including elastic replication, duplication, addition or join of services; and (ii) *Scale-down*, if the approach supports a shrinking phase, including deletion, removal, drop, eviction, merge or consolidation of services.
- *Constraints*: We use the following notations. (RS) resource constraint including capacity, processing capability/rate, CPU/GPU/memory utilization; (NT) network constraint including bandwidth usage, time or latency tolerance, service deadline; (PX) proximity constraint including hop-count, clustering, neighborhood locality, connectivity radius, maximum distance, coverage, etc.; and (BD) budget constraint

Table 1 – Comparison of proposed ARPA with existing work.

Work	Environment	Decentralized	Dynamic	Migration	Placement Adaptation		Constraints	Objectives
					Scale-up	Scale-down		
[29]	Other	✗	✗	✗	✗	✗	N/A	QoS
[57]		✓	✓	✓	✓	✓	N/A	CT, QoS
[58]		✓	✓	✗	✓	✓	RS	QoS
[59]		✓	✓	✓	✗	✗	N/A	CT, QoS
[7]	Cloud	✗	✗	✗	✗	✗	NT, RS	CT
[12]		✗	✗	✗	✗	✗	NT, RS	CT, QoE
[14]		✗	✗	✗	✗	✗	NT, RS	CT, QoS
[37]		✗	✗	✗	✗	✗	NT, RS	CT, QoE
[42]		✗	✓	✗	✗	✗	RC	CT
[60]		✓	✓	✓	✓	✓	N/A	CT, QoS
[21]	Edge	✓	✓	✓	✓	✓	N/A	CT, QoS
[39]		✗	✓	✓	✗	✗	BD, NT, RS	CT, QoE
[40]		✗	✓	✗	✗	✓	NT, RS	QoS
[44]		✗	✓	✓	✗	✗	NT, RS, PX	CT, LB, QoS
[49]		✗	✗	✗	✗	✗	NT, RS	EN
[51]		✗	✗	✗	✗	✗	BD, NT, RS PX	LB, QoS
[52]		✗	✗	✗	✗	✓	NT, RS, PX	CT, QoS
[53]		✗	✓	✓	✗	✗	RS, NT, BD	CT, QoE
[54]		✗	✗	✗	✗	✗	BD, PX	CT, QoS
[55]		✗	✗	✗	✗	✗	N/A	CT, QoS
[56]		✗	✗	✗	✗	✗	NT, RS PX	CT
[41]	Fog	✗	✗	✗	✗	✗	NT, RS	CT
[43]		✗	✗	✗	✗	✗	RS, NT	CT
[45]		✗	✓	✓	✗	✗	NT, RS	QoE, QoS
[46]		✓	✗	✗	✗	✗	BD, NT, RS, PX	QoE
[47]		✓	✓	✓	✗	✗	RS	EN, QoS
[48]		✓	✓	✓	✓	✓	RS	EN, QoS
[50]		✗	✓	✗	✗	✓	NT, RS	CT
ARPA		✓	✓	✓	✓	✓	NT, PX	CT, QoS

including fixed budgets, maximum cost, maximum number of servers/replicas.

- *Objectives*: We differentiate between the following notations. (CT) cost including provision, deployment, migration, execution, or storage cost minimization, profit maximization, or resource fee/wastage minimization; (LB) load balancing; (EN) energy consumption minimization; (QoE) Quality-of-Experience maximization concerning user, app or game-centric preferences; and (QoS) Quality-of-Service maximization including accessibility/availability, synchronization/consistency and fairness of services, routing optimization, resilience to failure, or bandwidth overhead minimization.

Obviously, there is a plethora of earlier work in cloud-only systems. Recently, however, a growing interest for efficient service placement at the FC and especially the EC layers is reported. Still, the vast majority refers to centralized approaches, which incur high computational overhead and are not ideal for the ever-changing conditions of multimedia cloud computing systems since they do not scale easily. The proposed ARPA aims at filling this literature gap with a placement solution that is both flexible and dynamic, by offering a completely distributed solution with adaptive relocation of the rendering services (via migration, replication and consolidation). Further, it acknowledges both user and provider-centric factors along with proximity principles found in FC ecosystems, to achieve an optimized trade-off between conflicting objectives that relate to QoS of the users on the one hand and cost-efficiency of the provider on the other hand.

3. SYSTEM ARCHITECTURE

The architecture behind the considered cloud/fog immersive multimedia service system is showcased in Fig. 1. It is comprised of three distinct layers as follows.

The top layer in Fig. 1 corresponds to the *cloud layer*. In immersive multimedia applications, e.g., virtual worlds, the virtual scene could comprise a vast open world. In order to compensate for the increased service demands by the high number of engaged users and manage the available resources effectively, distributed computing solutions are frequently adopted. Consequently, the core of the cloud layer consists of *multimedia engine* instances, that are geographically dispersed and hosted in various regions, in order to provide the users with high-end multimedia applications and virtual world state updates. Actually, the multimedia engines may be even further divided into multiple VMs, balancing the computing tasks

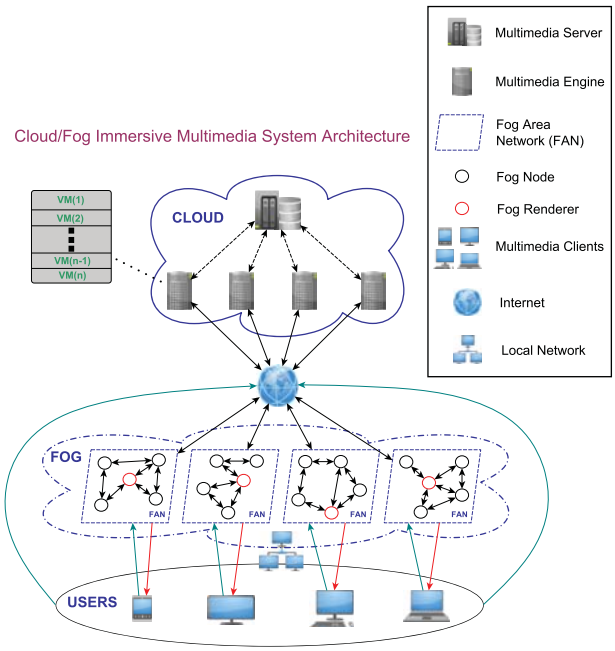

Fig. 1 – An example of the considered three-layered cloud/fog immersive multimedia system architecture.

and offering increased QoS through resource virtualization. Besides, they are teamed up through connection with a central *multimedia server*, which communicates with the provider's data center and database, and holds the AI entity, in a typical cloud fashion that allows their interconnection, as well as facilitates interactions between avatars (the users' playable characters-profiles) and other immersion-related elements. The latter is exceedingly important, since users expect to experience a single, large, and unfragmented virtual world allowing for seamless interactions among them in a transparent manner and without in-between loading lags.

Despite the preceding semi-centralized setup, the computation of immersive cloud multimedia environments has a very high demand for server capacities and remains subject to strict delay and traffic load requirements [61]. For example, consider the case where a multimedia engine is getting overwhelmed due to a spike in activity caused by a sudden virtual event taking place, such as a virtual festival. As a result, the multimedia engine may be led to a state, where it can no longer sufficiently handle the generated workload, leading to unacceptable performance degradation and loss in QoS, compromising the integrity of the whole system. The novelty behind the layering in Fig. 1 is the introduction of the intermediate *fog layer*, which leverages the FC paradigm, unburdening the multimedia engines and assisting in 3D frame rendering.

In particular, the fog layer includes various FC networks, established on various MANs, benefiting from existing infrastructure without additional capital expenses. These networks, henceforth, called *Fog Area Networks* (FAN)s, form an overlay between the end users' clients and their assigned cloud multimedia engine, containing devices, hereafter, named *fog nodes*, that possess capabilities similar to those of their associated multimedia engines, but of less computational and networking capacity, and are in close proximity to the end users. As such, they may vary in nature (e.g., idle PCs, VMs, base stations, smartphones, IoT devices, access points, etc.), however, they all share the potential of becoming *fog renderers*, by hosting an appropriate *rendering service*, that vastly reduces the downstream communication delay, since all rendering takes place near the immediate vicinity of the corresponding clients. The deployed fog nodes are illustrated as black-outlined circles in the constructed FANs in Fig. 1, whereas the ones that are eventually appointed the role of fog renderers are depicted as red-outlined circles respectively.

Lastly, the bottom part of the architecture encases the *users layer*. The proliferation of IoT and CC services along with the wide spread of various smart and personal mobile devices have enabled the users to engage with immersive multimedia technologies through diversified platforms and operating systems. In this way, any device can potentially host a *multimedia client* and allow the users to immerse in virtual worlds, freeing them from the necessity of owning specialized hardware/software, e.g., powerful CPUs/GPUs. The clients are connected through their local area connectivity to their closest fog node in their respective FANs, the latter acting as a gateway for connecting to the appropriate cloud engine and fog renderer (either itself or another fog node hosting a rendering service), continuously sending flows of data regarding their application demands and activity. Accordingly, virtual world processing and updating occur in the remote cloud, while the 3D scene rendering takes place at the fog renderers, reducing the distance the newly rendered frames need to travel and subsequently decreasing the communication delay.

The considered cloud/fog three-layered multimedia system is capable of capturing various network configurations, in the sense that even in cases where FANs do not exist between the top and bottom layers, the users' clients can still connect directly to the cloud, in which case the renderer will be instantiated at the most appropriate VM location inside the multimedia engines (that minimizes the downstream delay). For the rest of the paper, without loss of generality, it will be assumed that all clients are connected to a particular FAN, which in turn can connect to the cloud multimedia engines' VMs with negligible time since the communication speed and bandwidth between them are assumed to be very high (especially when considering the rise of 5G networks). However, a serious question that arises relates to the number and positioning of the fog renderers inside the FANs, that host the given rendering service, having always in mind the augmentation of QoS for the users and the reduction of the capital costs for the providers. This problem, in similar CC scenarios, has been typically addressed in related literature (e.g., [62]) as an instance of the FLPs [20].

4. FOG AREA NETWORK DESCRIPTION

In the current section, a formal description of the FAN is provided considering a typical MAN.

4.1 FAN topology

Each FAN's topology, as illustrated in the fog layer of Fig. 1, can be represented by a connected undirected graph, denoted as $G(V, E)$, where V is the set of fog nodes and E corresponds to the set of links between them. Then, any two fog nodes, $u, v \in V$, are considered connected if there exists a link $e(u, v) \in E$ between them. All fog nodes are assigned a maximum communication range, henceforth denoted as ξ. Subsequently, for $e(u, v)$ to exist, the positions of u, v must lie within ξ, or equivalently

$$w(u, v) \leq \xi, \quad \forall u, v \in V, \tag{1}$$

where $w(u, v)$ denotes the distance between u and v. Then, for a particular u ($u \in V$), let $\delta(u)$ denote the set of neighbor nodes of u, where $w(u, v) \leq \xi$ holds $\forall u, v \in V$, and let $N \triangleq \max_{\forall u \in V} |\delta(u)|$.

Besides, each link $e \in E$ is assigned a positive link *weight*, denoted as $d(u, v)$. Note that the physical meaning of this weight can vary depending on the case investigated (e.g., energy consumption, bandwidth consumption, etc.). In the current work, it represents the one-way[1] *communication delay* incurred by the data exchange between fog nodes u and v ($u, v \in V$), which can be obtained by corresponding measurement solutions such as the IETF RFC 7679 [63]. Then, it is derived that for any fog nodes $x, y \in V$, where the x and y are not adjacent ($y \notin \delta(u)$), the *total communication delay* $D(x, y)$ is equal to the concatenation of all delay weights along a *shortest path* ($\mathcal{SP}$) connecting the two nodes, which can be easily obtained via standard routing protocols or algorithms, such as "Dijkstra". Without any loss of generality, let $D(x, y) = D(y, x)$, while for the special case where $y = x$, i.e., for the same fog node x, the communication delay becomes $D(x, y) = D(x, x) = d(x, x) = 0$. Finally, let $k_{x,y}$ express the number of hops (i.e., the number of e links) separating x from y along the $\mathcal{SP}$ that connects them.

Assuming that time is slotted over a *finite horizon* $\tau = \{0, \dots, T\}$ and, thus, divided into discrete and equal-sized *time slots*, denoted as t ($0 \leq t \leq T$), let $F_s^t \subseteq V$ define a subset of fog renderers in G, i.e., fog nodes hosting a rendering service, denoted here as s, during the specific t. For the sake of simplicity, it is assumed that all fog nodes v ($v \in V$) can potentially become a renderer. Let v_s denote the case where fog node v has indeed become a fog renderer i.e., $v_s = \{v \in F_s^t, \forall t \in \tau\}$.

4.2 FAN definitions

It is common practice in network topologies such as the one presented previously to regard users' equipment, i.e., the client devices in current work, as not capable of hosting a service, due to battery constraints (e.g., mobile devices) or computational resource limitations. Although the considered topology model can be easily extended to include these, by incorporating the EC paradigm, further investigation into this aspect is beyond the scope of the current work.

Despite the aforementioned adopted limitation, the data generated at each fog node, containing interaction commands, is suitably mapped to the traffic load imposed by their assigned users (located at the users' layer). It is hypothesized that all users are engaged in the same virtual world, and so all information is transmitted in the form of equal-sized data packets in the FAN. This simplification is adopted in order to avoid the necessity and complexity of producing different frame sizes based on each client's perspective (e.g., 2D or 3D views, different resolutions, etc.). However, the data packets generation rate is heavily dependent on the user's immersion activity and so it can significantly vary from client to client. For example, a client x, actively partaking with 3D content creation in a virtual exhibition, generates data packets at its associated fog node $u \in V$ at a much higher rate than a different client y, witnessing the same scene as a passive spectator, at his fog node $v \in V$ respectively. This assumption is also easily extended to capture the application demands of users immersed via different types of devices, such as conventional desktop configurations or modern xR head-mounted displays. Likewise, let $r^t(u)$ define the mean rate, at which data packets, containing the information for the next virtual scene state update, are transferred through the network between fog node u and a given fog renderer v_s per time slot. In other words, $r^t(u)$ will be referred to as the multimedia service *rendering demands* of fog node u per t.

If multiple fog renderers exist in G, i.e., $|F_s^t| > 1$, each fog node u is served at time slot t by the closest (in terms of total communication delay) renderer, denoted here as $v_{s \to u}^t$, where $v_{s \to u}^t = \{v_s : D(u, v_s) \leq D(u, v_s'), \forall v_s' \in F_s^t\}$. Consequently, given an established $\mathcal{SP}$ between any fog node $u \in V$ and any renderer $v_s \in F_s^t$, then each v_s serves at t all nodes u for which $v_{s \to u}^t = v_s$ as defined earlier. Actually, at time slot t all nodes u and their assigned renderer v_s belong to the same subgraph, which in turn is mapped to a *shortest path tree*, denoted as $S_{v_s}^t$, rooted at $v_{s \to u}^t$ (where $v_{s \to u}^t = v_s$). Fig. 2(c) depicts three such subgraphs corresponding to a set F_s^t of three fog renderers (illustrated with different colors) after the creation of the respective shortest path trees based on the communication delay weights assigned to each link.

Besides, it is evident from Fig. 2 that $\forall u \in S_{v_s}^t$ there also exists a *subtree* $\check{S}_{v_s:u}^t$ (i.e., $\check{S}_{v_s:u}^t \subseteq S_{v_s}^t$) rooted at node u. Based on the aforementioned and the fact that each node u accumulates service rendering demands from all other nodes in $\check{S}_{v_s:u}^t$, then the *aggregate rendering demands* of fog node u at time slot t (i.e., its workload) are denoted as

[1] Note, that the Round-Trip Time, measured by probing packet tools or ping mechanisms, could also be used in this context without violating the general model.

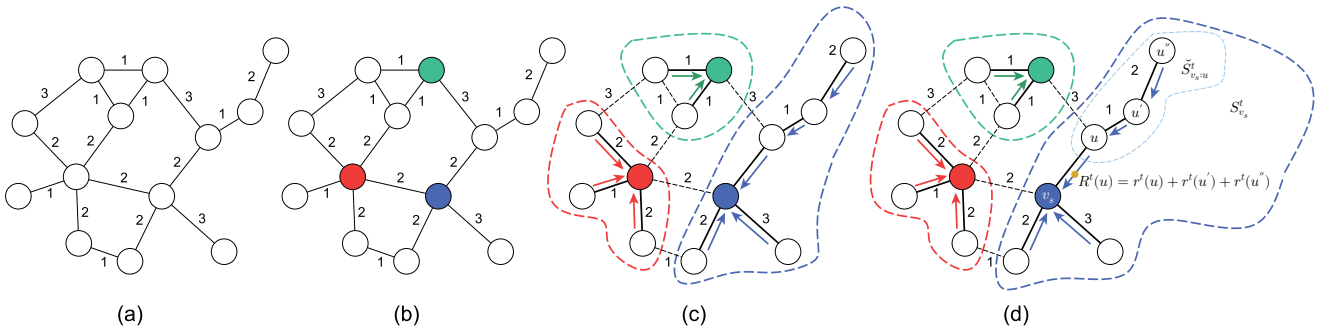

Fig. 2 – An example of shortest-path trees to thee renderers in a given FAN graph. The weights on the edges correspond to the communication delay. The bold lines represent the trees' paths. The arrows show the demand flow paths from all nodes u to their respective fog renderer $v_{s\to u}^t$ at time step t. In particular: (a) the FAN is defined; (b) three fog nodes are selected to become renderers; (c) the $\mathcal{SP}$ trees to the renderers are formed based on the communication delay weights of the links; and (d) the rendering demands $\forall u \in V$ are forwarded to the corresponding renderers $v_s \in F_s^t$ over the appropriate $S_{v_s}^t$, where each $u \in S_{v_s}^t$ accumulates rendering demands from all other nodes in each own subtree $\check{S}_{v_s:u}^t \subseteq S_{v_s}^t$.

$R^t(u)$, where

$$R^t(u) = \sum_{\forall v \in \check{S}_{v_s:u}^t} r^t(v). \qquad (2)$$

Fig. 2(d) depicts such a case for node $u \in V$, which agglomerates rendering demands from itself and its subtree $\check{S}_{v_s:u}^t$, which also contains fog nodes $u', u'' \in V$, thus $R^t(u) = r^t(u) + r^t(u') + r^t(u'')$ in this case.

5. PROBLEM FORMULATION

As already discussed, the optimal placement of services within a network has been typically tackled as an instance of the FLP [20]. To simplify the notations, the FLP problem formulated in the current section takes into account the presence of one type of multimedia service, simply referred to as the *rendering service*. The scenario of multiple rendering service types (e.g., based on the genre of different cloud games or the category of xR applications), that may simultaneously exist in the FAN, can be easily modeled by applying the same formulation for each different service type.

5.1 Cost model

Based on the preceding definitions we now present the cost model of the system that will drive the placement in the next sections.

5.1.1 *Service render cost*

As already stated, for every t each fog node $u \in V$ introduces to the FAN (on average) $r^t(u)$ rendering demands (i.e., additional workload), which must travel distance $\{u, \dots, v_s\}$ over an established $\mathcal{SP}$ to reach the overseeing v_s responsible for offering the s. Nevertheless, based on the definition of the total communication delay, the $D(u, v_s)_{\forall u \in S_{v_s}^t}$ is accordingly translated to a *service render cost* incurred to a particular fog node u in the FAN when being served by a particular fog renderer v_s during t (i.e., $u \in S_{v_s}^t$), which is proportional to its rendering demands, i.e.,

$$C_{ren}^t(v_{s\to u}) = r^t(u) D(u, v_s). \qquad (3)$$

However, one must keep in mind that past research has indicated that immersive applications have low tolerances on network delays [64]. Therefore, it is imperative to enforce specific delay thresholds, based on the multimedia service s provided by the system, in order to offer acceptable multimedia-oriented QoS. Let $L_s > 0$ denote this *upper latency bound* for each pair of fog node-renderer during the end of the placement procedure, and hence we end up with

$$L_s \geq C_{ren}^T(v_{s\to u}), \quad \forall u \in V, \forall v_s \in F_s^T. \qquad (4)$$

5.1.2 *Service access cost*

Given Eq. *(3)*, we can now go a step further and define the *service access cost* incurred to the FAN by a specific renderer during t, denoted here as $C_{acc}^t(v_s)$. In fact, since $r^t(u)$ rendering demands are generated per time slot t at each fog node $u \in V$, they contribute on average $r^t(u)D(u, v_s)$ additional render cost, and therefore the $C_{acc}^t(v_s)$ at time slot t is ultimately given by,

$$C_{acc}^t(v_s) = \sum_{\forall u \in S_{v_s}^t} C_{ren}^t(v_{s\to u}) = \sum_{\forall u \in S_{v_s}^t} r^t(u) D(u, v_s). \qquad (5)$$

5.1.3 *Service host cost*

Clearly, in order for the users to acquire maximum QoS from the cloud/fog immersive multimedia system the overall service access cost $C_{ren}^t(v_{s\to u})$ must be kept as low as possible $\forall u \in V$ and certainly below the L_s upper limit. Previous works (like [15]) have indicated that this is achievable by deploying more fog renderers. However, the careful placement of these renderers must be taken

into account since, for any new renderer that is instantiated, an additional *service host cost* for successfully hosting the particular rendering service (i.e., service installation, GPU configuration, maintenance, etc.), denoted here as C_s^{host}, is inflicted on the provider.

5.2 Problem statement

Given the preceding formulation and an initial arbitrary F_s^0 placement of renderers, the FLP approach here takes the form of distributedly discovering an F_s^T, that effectively minimizes *overall service provision cost* for the users and *overall service deployment cost* for the provider, abiding by the constraints of the FAN ecosystem and the offered multimedia applications. As mentioned in Section 1, this problem is here referred to as the "Latency Bounded Uncapacitated Fog Renderer Location" (LB-UFRL) problem, which falls under the category of the complex UFL problems.

Let C_{prov}^t and C_{depl}^t denote the two costs respectively. Then, it is derived that for a given placement of renderers F_s^t at any time slot t, the C_{prov}^t imposed on the system for rendering service provision, is suitably computed as the summation of all service access costs for accessing all $v_s \in F_s^t$, or alternatively,

$$C_{prov}^t = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S_{v_s}^t} C_{ren}^t(v_{s \to u}). \tag{6}$$

Likewise, the C_{depl}^t, at any time slot t, equates to,

$$C_{depl}^t = \sum_{\forall v_s \in F_s^t} C_s^{host} = |F_s^t| C_s^{host}. \tag{7}$$

Eventually, the optimization goal of the LB-UFRL at the end of the time horizon (i.e., $t = T$) becomes the minimization of the sum of the two respective cost functions. Let C^t denote the result of this summation, i.e., the *total cost* of the FAN taking into account all previously prescribed costs, i.e.,

$$C^t = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S_{v_s}^t} C_{ren}^t(v_{s \to u}) + \sum_{\forall v_s \in F_s^t} C_s^{host}. \tag{8}$$

Then, the problem is defined as:

$$\begin{aligned} &\textbf{LB-UFRL:} \quad \min\{C^T := C_{prov}^T + C_{depl}^T\}, \\ &\text{s.t.} \qquad\qquad\qquad\qquad \text{Eq. } (1),\ (4). \end{aligned} \tag{9}$$

6. LOCALIZED RELOCATION ANALYSIS

Because of its high complexity ($\mathcal{NP}$-hard for general graphs as a special case of the UFL), especially in terms of scalability and elasticity, computing an optimal feasible solution to the LB-UFRL could prove intractable, even with vast computing resources available. In fact, traditional centralized approaches, requiring global topology knowledge, are often deemed unsuitable for this task due to the sometimes extreme computational expenses. Thus, an elastic decentralized approach is designated hereinafter, utilizing solely local information, to dynamically relocate the services within the FAN, until the optimal renderers' location is discovered given an initial arbitrary placement.

After relocations do occur, the renderers' placement changes for the next time slot ($F_s^t \neq F_s^{t+1}$), and so does the underlying routing scheme, attributed to the new $\mathcal{SP}$s that are formed, since every fog node $u \in V$ will always prefer its closest in terms of delay renderer; that is, where $D(u, v_s') \leq D(u, v_s), \forall v_s \in F_s^t, \forall v_s' \in F_s^{t+1}$ (the equality holds for those $\mathcal{SP}$s that are not affected, i.e., when $v_s' = v_s$). Because it is impossible in an online and purely localized approach to *a priori* have global knowledge of the new $\mathcal{SP}$s that will be created via the relocation, our focus hereafter will be on the worst-case scenario where the only nodes that are impacted are the ones belonging to the subtrees of the participating renderers. A list of lemmas and theorems follows that designates the relocation behavior of the renderers, based on local network knowledge regarding the immediate neighborhood $\delta(v_s) \cap S_{v_s}^t$ ($\forall v_s \in F_s^t$), during a specific t, hypothesizing that the relocation will happen *instantaneously* during the same t without altering the underlying $\mathcal{SP}$s.

6.1 Service migration

Service *migration* has been proposed in past literature including cloud environments, with the aim of moving services (e.g., VMs) to the best locations within the network that minimize some objective function (e.g., [59]). Consider the case of one rendering service located at renderer $x_s \in F_s^t$, that needs to be migrated to some fog node $y \in \delta(x_s)$ during t. The challenges that naturally arise are twofold: (i) whether a *service migration* to fog node y, denoted as ${}^t\overrightarrow{x_s : y}$, will indeed incur a reduction to C^t for the next time slot, i.e., for $t = t + 1$; and (ii) which minimum condition should be satisfied to achieve this result based solely on local information available to x_s during t regarding its neighborhood $\delta(x_s)$.

Let $\Delta C({}^t\overrightarrow{x_s : y})$ denote the *post-migration cost difference* that is attributed to the migration of the rendering service from current renderer x_s to fog node $y \in \delta(x_s) \cap S_{x_s}^t$, at t, and let $\Delta C({}^t\overrightarrow{x_s : y})_{min} = \min\{\Delta C({}^t\overrightarrow{x_s : y}), \forall y \in \delta(x_s) \cap S_{x_s}^t\}$ be the *minimum post-migration cost difference* respectively.

Lemma 1. *Given any migration ${}^t\overrightarrow{x_s : y}$, where $y \in \delta(x_s) \cap S_{x_s}^t$ and $d(x_s, y) > d(x_s, z), \forall z \in \delta(x_s) \cap S_{x_s}^t$, there exists a minimum post-migration cost difference given by $C^t = C^{t+1} + \Delta C({}^t\overrightarrow{x_s : y})_{min}$, where*

$$\Delta C({}^t\overrightarrow{x_s : y})_{min} = C_{acc}^t(x_s) - C_{acc}^t(y). \tag{10}$$

Proof. The proof is included in Appendix A.1. □

The aim next is to determine under which minimum condition the $\Delta C({}^t\overrightarrow{x_s : y}) > 0, \forall y \in \delta(x_s) \cap S_{x_s}^t$ is satisfied, or similarly under which minimum condition $C^{t+1} < C^t$.

Theorem 1. *For any migration ${}^t\overrightarrow{x_s : y}$, where $y \in \delta(x_s) \cap S^t_{x_s}$, to result in a total cost reduction, there must at least exist a post-migration cost difference, given by $C^t = C^{t+1} + \Delta C({}^t\overrightarrow{x_s : y})$, where*

$$\Delta C({}^t\overrightarrow{x_s : y}) \geq 2R^t(y) - R^t(x_s) > 0. \quad (11)$$

Proof. The proof is included in Appendix A.2. □

In view of Theorem 1 it is clear that in order to achieve total cost reduction by ${}^t\overrightarrow{x_s : y}$ it is not necessary to have global network knowledge. In fact, it is sufficient enough to have local knowledge regarding the aggregate rendering demands of the x_s renderer's neighbor fog nodes y.

6.2 Service replication

Similar to [21], service *replication* corresponds here to instantiating a new renderer replica in a neighbor node given some local conditions that necessitate the existence of more services in order to cope with the rendering demands of the fog nodes. Evidently, the replication of services will have an impact on the underlying routing scheme, which should acknowledge the new renderer's location, since the new location will enforce fog nodes to forward their demands to the new renderer if their in-between total communication delay is less than previously. Besides this positive implication on the C^t_{prov}, there also exists a negative outcome on the total C^t_{depl} which is expected to rise with the addition of the extra host cost imposed by the new renderer. Hence, a judicious replication should be aware of the C^{host}_s and sufficiently compensate it by the cost reduction gain.
Let ${}^t\overleftarrow{x_s : y'}$ denote the *service replication* from renderer x_s to some fog node y' during time slot t. Also, let $\Delta C({}^t\overleftarrow{x_s : y'})$ denote the *post-replication cost difference* that can be achieved due to the specific rendering replica creation at t, and let $\Delta C({}^t\overleftarrow{x_s : y'})_{min} = \min\{\Delta C({}^t\overleftarrow{x_s : y'}), \forall y' \in \delta(x_s) \cap S^t_{x_s}\}$ be the *minimum post-replication cost difference* respectively. The new questions that are posed now are: (i) whether service replication can incur a sufficient reduction to C^t for the next time slot $t = t + 1$, that effectively compensates for the increased C^{t+1}_{depl}; and (ii) which minimum condition should be satisfied to obtain this result based solely on local information.

Lemma 2. *Given any replication ${}^t\overleftarrow{x_s : y'}$, where $x_s \in F^t_s$ and $y' \in V$, there exists an overall service provision cost reduction during the next time slot, i.e., $C^t_{prov} > C^{t+1}_{prov}$, as well as an overall service deployment cost increment, i.e., $C^t_{depl} < C^{t+1}_{depl}$.*

Proof. The proof is included in Appendix A.3. □

With Lemma 2 in mind, our goal now is to determine the condition to achieve total cost reduction after any replication. Since this requires global knowledge, we will focus on the minimum acceptable condition assuming that replication will occur locally, i.e, $y' \in \delta(x_s) \cap S^t_{x_s}$.

Theorem 2. *For any replication ${}^t\overleftarrow{x_s : y'}$, where $y' \in \delta(x_s) \cap S^t_{x_s}$, to result in total cost reduction, there must at least exist a post-replication cost difference given by $C^t = C^{t+1} + \Delta C({}^t\overleftarrow{x_s : y'})$, where*

$$\Delta C({}^t\overleftarrow{x_s : y'}) \geq R^t(y')d(y', x_s) - C^{host}_s > 0. \quad (12)$$

Proof. The proof is included in Appendix A.4. □

In view of Theorem 2 it is clear that in order to achieve total cost reduction by ${}^t\overleftarrow{x_s : y'}$ it is not necessary to have global network knowledge. In fact, it is sufficient enough to have local knowledge regarding the aggregate rendering demands of the x_s renderer's neighbor fog nodes y' and the communication delay weights of its own links.

6.3 Service consolidation

Due to the high dynamicity of the cloud/fog immersive multimedia systems, there might be cases where the number of instantiated rendering services is redundant. Thus, a prudent placement strategy should be able to dynamically reduce the number of renderers, if deemed necessary, in order to reduce the overall capital expenses for the multimedia provider [60]. The number of services in the FAN is reduced by merging two fog renderers through a process named service *consolidation*. The rationale behind consolidation is to check whether the merge would decrease the overall deployment cost, which subsequently could lead to a total cost decay. After consolidation occurs, only one of the participating renderers is left in the FAN. However, we must keep in mind that this will have a negative impact on the overall provision cost due to longer distances between the merged renderer and the fog nodes.
Let ${}^t\overleftarrow{x_s : y''_s}$ denote the *service consolidation* from renderer y''_s to some fog renderer x_s during time slot t. Let $\Delta C({}^t\overleftarrow{x_s : y''_s})$ denote the *post-consolidation cost difference* that can be achieved due to the specific service merge at t, and let $\Delta C({}^t\overleftarrow{x_s : y''})_{min} = \min\{\Delta C({}^t\overleftarrow{x_s : y''}), \forall y'' \in \delta(x_s) \cap S^t_{x_s}\}$ be the *minimum post-consolidation cost difference*, respectively. We now are called to answer: (i) whether service consolidation can incur a sufficient reduction to C^t for the next time slot $t = t + 1$, that effectively compensates for the increased C^{t+1}_{prov}; and (ii) which minimum condition should be satisfied to obtain this result based solely on local information.

Lemma 3. *Given any consolidation ${}^t\overleftarrow{x_s : y''_s}$, where $x_s, y''_s \in F^t_s$, there exists an overall service deployment cost reduction during the next time slot, i.e., $C^t_{depl} > C^{t+1}_{depl}$, as well as an overall service provision cost increment, i.e., $C^t_{prov} < C^{t+1}_{prov}$.*

Proof. The proof is included in Appendix A.5. □

Via Lemma 3, it is straightforwardly deduced that proving when consolidation can indeed achieve total cost reduction will require global topology knowledge. Thus, our focus is shifted on the minimum acceptable condition for

$C^t > C^{t+1}$, assuming that consolidation will occur locally, i.e, $y''_s \in \delta(x_s)$.

Theorem 3. *For any consolidation ${}^t\overleftarrow{x_s : y''_s}$, where $y''_s \in \delta(x_s)$, and $d(y''_s, x_s) > d(y''_s, v_s), \forall v_s \in F^t_s \cap \delta(y''_s)$, to result in total cost reduction, there must at least exist a post-consolidation cost difference given by $C^t = C^{t+1} + \Delta C({}^t\overleftarrow{x_s : y''_s})$, where*

$$\Delta C({}^t\overleftarrow{x_s : y''_s}) \geq C^{host}_s - R^t(y'')d(y''_s, x_s) > 0. \quad (13)$$

Proof. The proof is included in Appendix A.6. □

In view of Theorem 3, it is clear that in order to achieve total cost reduction by ${}^t\overleftarrow{x_s : y''_s}$, it is not necessary to have global network knowledge. In fact, it is sufficient enough to have local knowledge regarding the aggregate rendering demands of the x_s renderer's neighbor fog renderers y''_s and the communication delay weights of its own links.

6.4 Upper latency compensation

The previous theorems provide us with minimum local conditions for relocating services with simultaneous total cost reduction. Nevertheless, they do not guarantee that by the end of the service placement process the delay constraint *(4)* will also be satisfied, since this is intimately connected to (i) the value of the upper latency bound L_s; (ii) the initial locations of the services; and (iii) the relocation sequence itself. Therefore, we would require global knowledge to efficiently address the issue. Given this limitation, we seek a local condition that will ultimately lead to the required result.

Intuitively, a way to compensate for the upper latency threshold would be to generate more service replicas that incur less render costs. Note, that for some renderer v_s the $C^t_{ren}(v_{s\to u})$ in Eq. *(3)* can be easily computed by each $u \in S^t_{v_s}$, which can then be forwarded to its parent node through the data packets (i.e., encapsulated inside $r^t(u)$) that are regularly being exchanged between them, even during inactive sessions. In fact, each fog node $u \in S^t_{v_s}$ along the formed $\mathcal{SP}$s, from the leaves to the root, i.e., the renderer, can (on a per hop basis) compare all render cost values arriving from its neighbor nodes u' in its respective subtree $\check{S}^t_{v_s:u}$, and determine the one whose own subtree reports the *highest service render cost*, denoted here as $\hat{C}^t_{ren}(u) = \max\{\hat{C}^t_{ren}(u'), \forall u' \in \check{S}^t_{v_s:u} \cap \delta(u)\}$. Eventually, v_s after monitoring all $\hat{C}^t_{ren}(u)$ of its own assigned neighbors can then determine per t the highest render cost of its whole subtree as $\{\hat{C}^t_{ren}(v_s) = \max\{\hat{C}^t_{ren}(u), \forall u \in S^t_{v_s} \cap \delta(v_s)\}$. Given $\hat{C}^t_{ren}(v_s)$, it is clear that for constraint *(4)* to be satisfied it is enough for

$$L_s \geq \hat{C}^T_{ren}(v_s), \quad \forall v_s \in F^T_s. \quad (14)$$

Based on this observation, and the fact that $\hat{C}^t_{ren}(v_s)$ can be locally available to any renderer v_s during a particular t, a service replication ${}^t\overleftarrow{v_s : \nu}$ is triggered when $\hat{C}^t_{ren}(v_s) = \hat{C}^t_{ren}(\nu)_{\forall \nu \in S^t_{v_s} \cap \delta(v_s)} > L_s$.

Lemma 4. *When constraint (4) is infringed for some node $u \in S^t_{v_s}$, the number of replications required to sufficiently compensate for L_s is upper bounded by k_{u,v_s}.*

Proof. The proof is included in Appendix A.7. □

Theorem 4. *At the end of the rendering service relocation, no fog node exists that violates L_s.*

Proof. The proof is included in Appendix A.8. □

7. ALGORITHM DESIGN

The preceding analysis motivates the proposal of an elastic distributed placement algorithm (i.e., the ARPA) for tackling the LB-UFRL. ARPA utilizes (per t) the previous relocation conditions to dynamically shift an arbitrary initial placement of renderers to optimal locations within the FAN. The details of ARPA are enclosed in Algorithm 1.

7.1 The proposed ARPA

Suppose that for $t = 0$ some fog nodes, located at some arbitrary initial locations in the FAN, have become fog renderers, i.e., $|F^0_s| \geq 0$. Given a finite time horizon (Line 1), the hosted rendering services are then allowed to be relocated (e.g., migrate, replicate, or merge) until an optimal placement is achieved adhering to the LB-UFRL constraints. To this end, each renderer (Line 2) in a distributed fashion, by monitoring its assigned one-hop neighbor fog nodes (Line 5), first checks whether a migration can be undertaken during the specific time slot. Otherwise, it continues to verify if the conditions for a replication, or subsequently for a consolidation of services, are satisfied. The aim is to minimize the sum of costs in the LB-UFRL, based strictly on local information.

The first relocation control in ARPA, i.e., Line 6, designates when an s migration takes place. From Theorem 1 it is clear that if the minimum condition $\Delta C({}^t\overrightarrow{x_s : y})_{min} = 2R^t(y) - R^t(x_s) > 0$ is satisfied, then a total cost reduction is achieved. Alternatively, if $2R^t(y) > R^t(x_s) \Leftrightarrow R^t(y) > \frac{R^t(x_s)}{2}$, then $C^t > C^{t+1}$ also holds.

Rule 1 (ARPA Migration). *Service migration ${}^t\overrightarrow{x_s : y}$ occurs if and only if $R^t(y) > \frac{R^t(x_s)}{2}$.*

Notice that if migration does occur at t (Line 7), then it is certain that it will lead to C^t reduction because $|F^{t+1}_s| = |F^t_s|$ and $C^{t+1}_{acc}(y_s) < C^t_{acc}(x_s)$, where $x_s \in F^t_s$ and $y \in S^t_{x_s} \cap \delta(x_s)$. In other words, for $t+1$ the service s has moved to the neighbor location with the highest impact on $C^t_{acc}(x_s)$ (over the half), reducing the overall delay cost for reaching the service in $S^{t+1}_{y_s}$ during $t+1$, i.e., $F^{t+1}_s = F^t_s \cup \{y\} \backslash \{x_s\}$ (Line 8). The particular property is conservative inasmuch there might be cases where $C^{t+1}_{acc}(y_s) < C^t_{acc}(x_s)$ holds, but the s movement may not occur, depending on the FAN's topology and the aggregate demands of the neighbor fog nodes. Nevertheless, for all $x_s \in F^t_s$, the service migration definitely leads to $C^t_{acc}(x_s)$ decay, hence, the C^t is also reduced.

Algorithm 1 ARPA

Input: FAN $G(V, E)$, Set of initial renderers F_s^0
Output: Subset $F_s^T \subseteq V$ where min C^T
Vars: L_s, T;

1: **while** $t \neq T$ **do**
2: **for** $\forall x_s \in F_s^t$ **do**
3: MIGRATION = FALSE; REPLICATION = FALSE;
4: CONSOLIDATION = FALSE;
5: **for** $\forall y \in S_{x_s}^t \cap \delta(x_s)$ **do**
6: **if** $R^t(y) > R^t(x_s)/2$ **then**
7: ${}^t\overrightarrow{x_s : y}$; ▷ Migration from x_s to y.
8: $F_s^{t+1} \leftarrow F_s^t \cup \{y\} \backslash \{x_s\}$;
9: MIGRATION = TRUE;
10: BREAK;
11: **end if**
12: **end for**
13: **if** !MIGRATION **then**
14: **if** $\hat{C}_{ren}^t(x_s) > L_s$ **then**
15: **for** $\forall y' \in S_{x_s}^t \cap \delta(x_s)$ **do**
16: **if** $\hat{C}_{ren}^t(x_s) = \hat{C}_{ren}^t(y')$ **then**
17: ${}^t\overleftarrow{x_s : y'}$; ▷ Replication (a) from x_s to y'.
18: $F_s^{t+1} \leftarrow F_s^t \cup \{y'\}$;
19: REPLICATION = TRUE;
20: BREAK;
21: **end if**
22: **end for**
23: **end if**
24: **if** !REPLICATION **then**
25: **for** $\forall y' \in S_{v_s}^t \cap \delta(x_s)$ **do**
26: **if** $R^t(y')d(y', x_s) > C_s^{host}$ **then**
27: ${}^t\overleftarrow{x_s : y'}$; ▷ Replication (b) from x_s to y'.
28: $F_s^{t+1} \leftarrow F_s^t \cup \{y'\}$;
29: REPLICATION = TRUE;
30: BREAK;
31: **end if**
32: **end for**
33: **end if**
34: **end if**
35: **if** !MIGRATION $\wedge$!REPLICATION **then**
36: **for** $\forall y_s'' \in F_s^t \cap \delta(x_s)$ **do**
37: **if** $R^t(y')d(y', x_s) < C_s^{host} \wedge \hat{C}_{ren}^t(y_s'') + R^t(y_s'')d(y_s'', x_s) \leq L_s$ **then**
38: ${}^t\overrightarrow{x_s : y_s''}$; ▷ Consolidation from y_s'' to x_s.
39: $F_s^{t+1} \leftarrow F_s^t \backslash \{y_s''\}$;
40: CONSOLIDATION = TRUE;
41: BREAK;
42: **end if**
43: **end for**
44: **end if**
45: **end for**
46: $t = t + 1$;
47: **end while**

As already stated, each renderer x_s has the capability of monitoring the highest service render costs $\hat{C}_{ren}^t(y')$ reported by its assigned neighbor fog nodes $y' \in S_{x_s}^t \cap \delta(x_s)$. When a renderer detects an upper latency violation in its $\mathcal{SP}$ tree (that is, an infringement of L_s according to constraint *(4)*), a service replication is triggered to address it (Line 17) by activating a new replica at the neighbor y' (belonging to its $\mathcal{SP}$ tree, i.e., Line 15) where $\hat{C}_{ren}^t(x_s) = \hat{C}_{ren}^t(y')$ (Line 16) according to the following rule.

Rule 2 (ARPA Replication (a))**.** *Service replication* ${}^t\overleftarrow{x_s : y'}$ *occurs if* $\hat{C}_{ren}^t(x_s) > L_s$*, where* $\hat{C}_{ren}^t(x_s) = \hat{C}_{ren}^t(y'), \forall y' \in S_{x_s}^t \cap \delta(x_s)$*, when Rule 1 is not valid.*

By instantiating a new replica at the $y' \in S_{v_s}^t \cap \delta(x_s)$ (Line 25), then it is ensured that the new placement during the next time slot will include y' (Line 18) resulting in $\hat{C}_{ren}^{t+1}(y_s') < \hat{C}_{ren}^t(x_s)$. This rule will repeat for at most k_{u,x_s} times, when $\hat{C}_{ren}^t(x_s) = C_{ren}^t(x_{s \to u})$, as revealed by Lemma 4. Interestingly, Rule 2 does not oppose the first rule of migration, since its execution depends on the prior satisfaction of Rule 1 (Line 13), meaning that a replication will be carried out if migration has not already occurred and only when an upper latency violation is observed (Line 14).
To further offer elasticity, each s is allowed to replicate to a new location during t when it is deemed more worthy to open a new renderer there, rather than forward the service demands over long distances, resulting in a reduced C^{t+1}. In fact, ARPA decides on this action by comparing the expected provision benefit with the deployment cost, i.e., the overall provision cost reduction should sufficiently compensate for the corresponding overall deployment cost increment in order for the total cost to be reduced. In view of Theorem 2, it is enough for $\Delta C({}^t\overleftarrow{x_s : y'})_{min} = R^t(y')d(y', x_s) - C_s^{host} > 0$ to be satisfied, thus ARPA's third relocation control (Line 26) allows for a service replication according to the sequel rule.

Rule 3 (ARPA Replication (b))**.** *Service replication* ${}^t\overleftarrow{x_s : y'}$ *also occurs if* $R^t(y')d(y', x_s) > C_s^{host}$*, when Rules 1 and 2 are not valid.*

Notice that Rule 3 does not contradict Rule 2 (and by extension Rule 1 is also not opposed), since it will be executed only when the latter has not been performed respectively (i.e., Line 24), and so long as the conditions are met. With the creation of the new replica (Line 27), y'' becomes part of the new placement for $t = t + 1$ (Line 28).
On the contrary, to lessen the number of deployed renderers (i.e., the overall deployment cost), ARPA checks whether a consolidation of services between x_s and some $y_s'' \in F_s^t \cap \delta(x_s)$ (Line 36) can be performed to reduce C_{depl}^t and in turn the C^t, ensuring firstly that the extra C_{prov}^{t+1} induced by the consolidation is sufficiently compensated, and secondly that the L_s upper bound will not be violated. For the first part, according to Theorem 3, for total cost reduction it is enough for $\Delta C({}^t\overleftarrow{x_s : y_s''})_{min} = C_s^{host} - R^t(y'')d(y_s'', x_s) > 0$, or similarly for $C_s^{host} > R^t(y'')d(y_s'', x_s)$. For the second part, we enforce that post any consolidation ${}^t\overleftarrow{x_s : y_s''}$, the additional access cost imposed by the longer distances and higher communication cost retains the $\hat{C}_{ren}^{t+1}(x_s)$ below the L_s limit (Line 37). Thus, the following rule is derived.

Rule 4 (ARPA Consolidation)**.** *Service consolidation* ${}^t\overleftarrow{x_s : y_s''}$ *occurs if and only if* $R^t(y')d(y', x_s) < C_s^{host}$ *and* $\hat{C}_{ren}^t(y_s'') + R^t(y_s'')d(y_s'', x_s) \leq L_s$*, when Rules 1, 2, and 3 are not valid.*

Notice that Rule 4 does not oppose any of the previous rules, since it will only be executed when no other rule applies (i.e., Line 35), and only if the conditions are

met. Thus, the renderers successfully merge (Line 38) taking advantage of strictly local information known to renderer x_s regarding the aggregate rendering demands and highest service render cost of y''_s, as well as their in-between communication delay. Besides, it ensures that when consolidation is completed the remaining renderer (i.e., the x_s) will still not violate the upper latency bound; that is, hypothesizing that $z \in S^t_{y''_s}$ is the node where $C^t_{ren}(y''_{s\to z}) \equiv \hat{C}^t_{ren}(y''_s)$, then $r^t(z) \leq R^t(y''_s)$ (the equality holds for the special case where only y''_s belongs to $S^t_{y''_s}$, or equivalently $|S^t_{y''_s}| = 1$) and so $\hat{C}^t_{ren}(y''_s) + r^t(z)d(y''_s, x_s) \leq \hat{C}^t_{ren}(y''_s) + R^t(y''_s)d(y''_s, x_s) \leq L_s$ is satisfied. After consolidation has been completed, for $t = t+1$ the previously active rendering service on fog node y'' is now switched off and removed from F^{t+1}_s (Line 39), concluding that $C^{t+1}_{depl} < C^t_{depl}$.

Lemma 5. *The time complexity of ARPA is $\mathcal{O}(|F^t_s|N)$, where $N = \max\{|\delta(u)|, \forall u \in V\}$.*

Proof. The proof is included in Appendix A.9. □

If none of the abovementioned rules are valid, then the service automatically stops relocating, meaning that the solution has been found. Contrary to this, if some network or topology change manifests that impacts the underlying $\mathcal{SP}$s or aggregate rendering demands (e.g., due to fog node mobility) then the process is again automatically resumed. This makes ARPA resilient to the high dynamicity of the cloud/fog immersive multimedia systems and appropriate for scaling up and down to meet the current demands.

Theorem 5. *The proposed ARPA for the LB-UFRL converges to a solution after a finite number of time slots.*

Proof. The proof is included in Appendix A.10. □

7.2 Discussion

There are some challenges posed by the design of ARPA that must be identified and are basically twofold. i) Its distributed nature may allow for more than one fog renderer to simultaneously attempt to access the same FAN resource through migration, replication, or/and consolidation. This limitation can be tackled with the careful synchronization of the nodes, through an appropriate network protocol, that will forbid events from taking place at the same time. ii) Depending on the FAN's density, the values of the upper latency bound and the service host cost, as well as the initial placement of rendering services, ARPA may not always reach the global optimal but instead provide a suboptimal solution which, however, closely approximates the optimum solution to the LB-UFRL. The latter most often occurs in circumstances where the C^t_{depl} decrease, L_s compensation and C^t_{prov} reduction seriously contradict one another, and thus achieving a thin balance between the three is extremely complex with purely localized topology knowledge. To minimize chances of halting in local minima, we conjecture that a relaxation of the LB-UFRL's constraints should take place. Alternatively, it could prove beneficial for the multimedia provider to decide at what point exactly to terminate the algorithm and what conditions should eventually be applied to the FAN in terms of rendering service allocation in order to achieve the best trade-off among the conflicting targets.

8. EVALUATION WITH SYNTHETIC DATA

In this section, ARPA is evaluated through a series of simulation experiments with synthetic data via the OMNeT++ V5.6.2[2] discrete simulator [65].

8.1 Simulation setup

To comprehensively evaluate the proposed ARPA, we construct various FANs comprised of random geometric graphs, following a normalized topology (1×1) of 100 fog nodes. Three values for the maximum connectivity radius are considered next, i.e., $\xi = \{0.20, 0.30, 0.40\}$. Likewise, a connection between any pair of nodes $u, v \in V$ exists only if $w(u, v) \leq \xi$. The rationale behind this approach is to mirror FANs of different densities, which however are expected to have relatively small sizes (e.g., within an existing MAN) with a limited number of fog nodes available. Further, $\forall u \in V$ a random value is assigned for their average service rendering demands rate, i.e., $r^t(u) \in [0, 1]$. The topology and rendering demands remain fixed throughout the simulation. All fog nodes can also host an s. Unless otherwise stated, three representative values are used for the host cost and upper latency bound; that is, $C^{host}_s = \{0.03, 0.06, 0.09\}$ and $L_s = \{0.04, 0.06, 0.08\}$ respectively. Estimations regarding the $R^t(u)$ and $\hat{C}^t_{ren}(u)$, $\forall u \in V$, take place considering a time slot of $t = 10$ seconds. Service data flows in the network towards the appropriate renderers over $\mathcal{SP}$s derived from *Dijkstra*'s algorithm, according to $D(u, v_s)$, per t. Each simulation terminates at $t = T = 1000$ seconds. An initial s is arbitrarily instantiated on a random fog node during the FAN construction, becoming the first fog renderer placement, i.e., $|F^0_s| = 1$. Different seeds are then used per experimental run to generate randomness for the network and topological parameters.

8.2 Simulation results

The simulation results presented next demonstrate qualitative aspects to confirm our analytical findings. The main motivation is to show how total cost reduction is achieved, while meeting the given constraints, when the ARPA is implemented under the different scenarios regarding the mentioned ξ, C^{host}_s and L_s cases. The approach is primarily to present the results of individual runs and reveal details of the ARPA's behavior that would not be visible via averaged results. Nevertheless, a comparison of averaged results against optimal solutions, measured by the IBM ILOG CPLEX Optimizer solver

[2] https://omnetpp.org/documentation

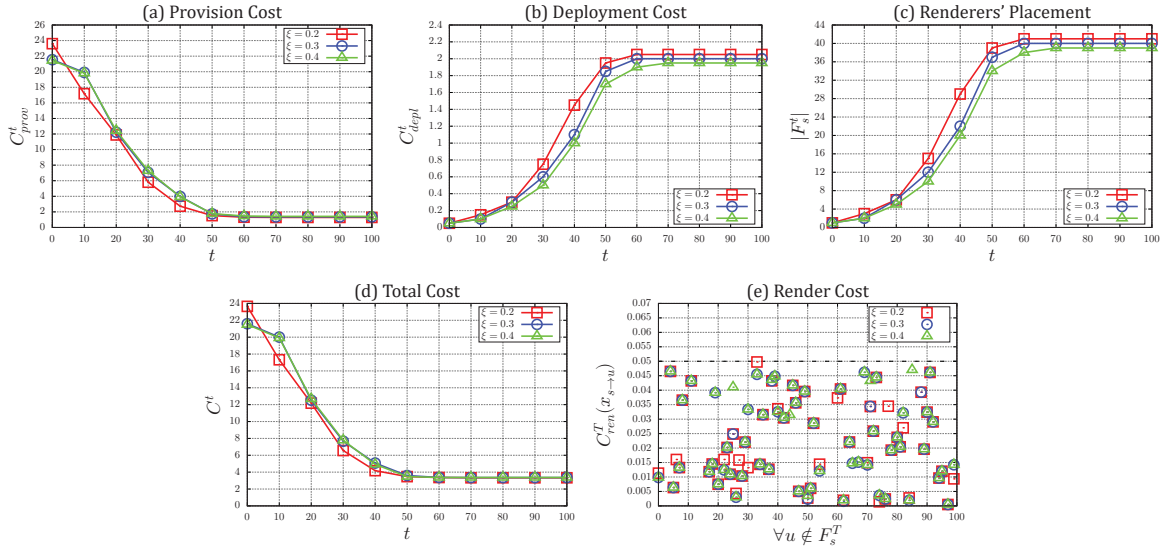

Fig. 3 – (a) C^t_{prov}, (b) C^t_{depl}, (c) $|F^t_s|$, and (d) C^t as a function of t, as well as (e) $C^T_{ren}(x_{s\to u}), \forall u \in V \backslash F^T_s$, with fixed $C^{host}_s = 0.05$ and $L_s = 0.05$, for three values of the connectivity radius ξ.

V20.1.0[3], is also presented separately in a sequel section, to give an overview of our proposed algorithm's approximation performance.

8.2.1 *Varying connectivity radius*

Initially, we explore the impact of the connectivity radius on ARPA's behavior. Fig. 3(a) graphically illustrates the provision cost as a function of t for the three considered values of ξ with fixed $C^{host}_s = 0.05$ and $L_s = 0.05$. Obviously, for each successive t, in all depicted cases, the C^t_{prov} decays until no more s relocations (i.e., migration, replication, or consolidation) can occur, at which point it stabilizes into a straight line. In contrast, Fig. 3(b) depicts the corresponding deployment cost C^t_{prov}. Initially, there is a spike in the creation of renderers in order to compensate for the L_s. Besides, the relatively low host cost allows for further replications, deeming more worthy to instantiate more renderers rather than forward demands over long distances with higher total communication delay. This is perfectly mirrored in Fig. 3(c). However, as time progresses the replication process is halted since consolidation of services takes over to reduce the C^t. Finally, migration leads the services to the most favorable locations. This relates to the conservatism of ARPA, which prohibits the opening of additional services based on the values of the host cost and the upper latency bound. Thus, the total cost C^t fast decreases until the optimized placement is finally reached at $t = 50$, i.e., Fig. 3(d). An important criterion for the efficiency of ARPA, is its ability to abide by the given latency threshold. Fig. 3(e) verifies the claim since it presents the final $C^T_{ren}(x_{s\to u}) \forall u \in V \backslash F^T_s$ (since all renderers have render cost equal to zero). Clearly, when ARPA terminates, the render cost of all nodes is below the L_s limit, which is depicted with a straight dotted line. Intuitively, ξ plays a significant role in $|F^T_s|$, since for denser networks the final placement is comprised of fewer renderers (due to the existence of more node links that lead to the formation of $\mathcal{SP}$s with lesser total communication delay), while the C^t remains close in all cases.

8.2.2 *Varying host cost*

To better contextualize how the host cost affects ARPA's execution, Fig. 4 plots the same variables as previously for fixed $\xi = 0.3$ and $L_s = 0.09$, regarding the three different values of C^{host}_s. Again, C^t_{prov} (in Fig. 4(a)) quickly drops as more renderers open to provide rendering services. Contrary to the preceding results, Fig. 4(b) shows the clear relation between C^t_{depl} and the $|F^T_s|$ (i.e., Fig. 4(c)), since for lower host cost a larger $|F^T_s|$ is allowed to exist in the FAN without negatively impacting the deployment cost. On the other hand, as the host cost increases renderers are increasingly prohibited from replicating often. Hence, the reduction in provision cost sufficiently compensates for the increment in deployment cost for all cases and therefore ARPA once more fast converges to a solution regarding the C^t, as observed by Fig. 4(d). Note that for $t = T$, no node exists that violates the L_s constraint. This is easily viewed in Fig. 4(e), which encases the render cost of all fog nodes (minus the renderers) for each case. In fact, for a large number of renderers (e.g., for $C^{host}_s = 0.03$) it is clear that the render cost is far lower than the given limit.

8.2.3 *Varying upper latency*

For the third simulation scenario, we vary the value of the upper latency bound, as depicted in Fig. 5, for fixed $\xi = 0.3$ and $C^{host}_s = 0.08$, since it is imperative to validate that ARPA is able to efficiently address the constraint under different situations, and result in locations

[3] https://www.ibm.com/docs/en/icos/20.1.0

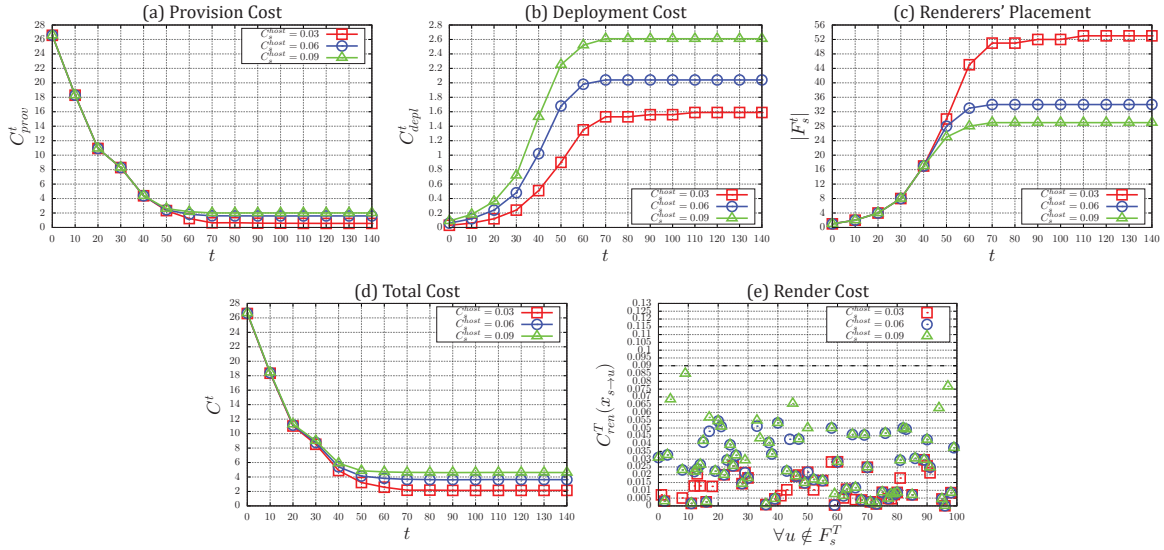

Fig. 4 – (a) C^t_{prov}, (b) C^t_{depl}, (c) $|F^t_s|$, and (d) C^t as a function of t, as well as (e) $C^T_{ren}(x_{s\to u})$, $\forall u \in V \backslash F^T_s$, with fixed $L_s = 0.09$ and $\xi = 0.3$, for three values of the host cost C^{host}_s.

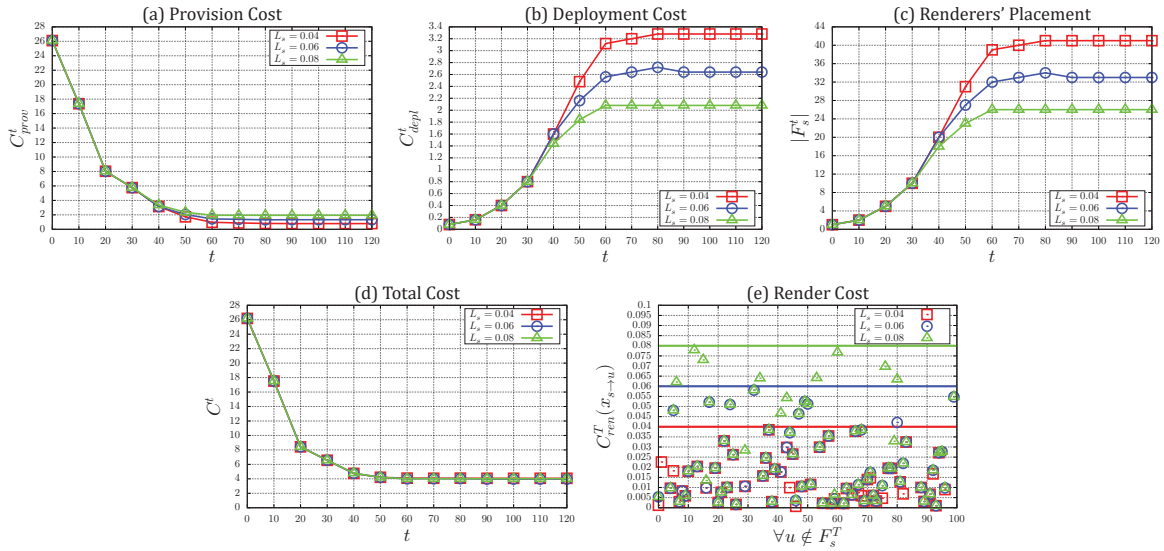

Fig. 5 – (a) C^t_{prov}, (b) C^t_{depl}, (c) $|F^t_s|$, and (d) C^t as a function of t, as well as (e) $C^T_{ren}(x_{s\to u})$, $\forall u \in V \backslash F^T_s$, with fixed $C^{host}_s = 0.08$ and $\xi = 0.3$, for three values of the upper latency threshold L_s.

that obey the render cost limitations. By Fig. 5(a) it is clear that once more the ARPA is able to minimize the provision cost independently of the latency bound. In fact, for all cases we observe an over 90% reduction in C^t_{prov} when the placement terminates. As shown in Fig. 5(b) and 5(c), the C^t_{depl} and $|F^t_s|$ follow the same distribution in all three cases since the placement for the first three t slots is dominated by the same servers relocating to the same FAN locations. However, for $t \geq 40$, the higher L_s values enforce the servers to merge or move towards the best locations faster, while lower ones compel ARPA to create more replicas, before migration and consolidation can take place, in order to address the render cost infringements. Thus, for $t \geq 100$ and $L_s = 0.08$, 26 renderers are enough to meet the requirements, whereas for $L_s = 0.06$ and $L_s = 0.04$, 33 and 41 renderers are demanded respectively. Still, ARPA manages to respect the corresponding thresholds (i.e., Fig. 5(e)), striving to place the servers at the most suitable locations that abide by these conditions, ultimately yielding total cost reduction. The last is clearly witnessed in Fig. 5(d) where the C^t monotonically decreases for each consecutive time slot and converges to a solution, minimizing the total cost by approximately 85% in all three cases.

8.2.4 Altering parameters

A crucial factor, as already stated, for the appropriateness of ARPA is its ability to conform the renderers' placement based on fluctuations in the network conditions due to

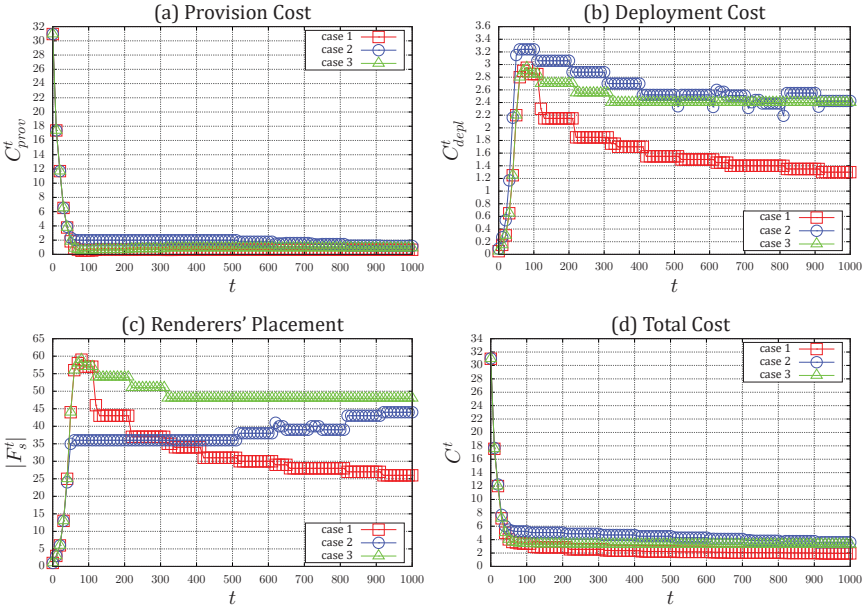

Fig. 6 – (a) C^t_{prov}, (b) C^t_{depl}, (c) $|F^t_s|$, and (d) C^t as a function of t with $\xi = 0.3$, for three cases where every 100 sec we re-initialize: 1) the $r^t(u) = [0, 1]$ and for fixed $C^{host}_s = 0.05$ and $L_s = 0.03$; 2) the initial $C^{host}_s = 0.09$ by subtracting 0.005 and for fixed $L_s = 0.07$; and 3) the initial $L_s = 0.03$ by adding 0.005 and for fixed $C^{host}_s = 0.05$.

the users' variable engagement activity. To capture this aspect, in Fig. 6 we plot the behavior of ARPA in respect to (a) C^t_{prov}, (b) C^t_{depl}, (c) $|F^t_s|$, and (d) C^t for three different experimental cases with fixed $\xi = 0.3$, where we alter for every 10 time slots, i.e., every 100 sec., (i) the demands, (ii) the host cost, and (iii) the upper latency. Apparently, in all cases the total cost once more is minimized in the first few time slots based on the initial conditions we have enforced, pausing in this way the execution of ARPA. From there, however, and depending on the alterations we make in the network, we can see that each time we re-initialize a parameter, ARPA's execution is automatically resumed in order to shift the placement to new locations that meet the new conditions and constraints. Actually, it takes a few time slots after each re-initialization to perform and adjust the placement in order to address the new problem. Because by default ARPA's rules relocate the renderers under the condition that the total cost under all circumstances is decreased, then, even if the conditions change, ARPA will still manage to tackle every new LB-UFRL achieving C^t reduction, as displayed in Fig. 6(d). Interestingly, for cases 1 and 3, the renderers' placement, i.e., $|F^t_s|$ in Fig. 6(c) is also reduced indicating that the number of servers instantiated during the initial placement is enough to address the alterations by just migrating or merging the services. In contrast, for case 2 where we iteratively reduce the C^{host}_s by 0.05, the impact on the C^t_{depl} is severe. Hence, more replications are allowed to occur to offer enhanced C^t_{prov} and thus $|F^t_s|$ rises. These results verify the expected behavior and showcase the capability of ARPA to remain unaffected by the changes and based on the placement of renderers to always adapt yielding optimized solutions.

8.2.5 Convergence to optimal solution

The aforementioned offered a microscopic overview of ARPA's behavior under different deployment scenarios. Yet, in order to better contextualize the approach and offer greater validity, a macroscopic overview is also deemed necessary to capture the approximation ability of ARPA towards the optimal solutions under different values for (i) the ξ, (ii) the C^{host}_s, and (iii) the L_s.

Let C_{OPT} denote the optimal solution of each LB-UFRL problem obtained by the optimizer. Then, the ratio $\frac{C^t}{C_{OPT}}$ declares how close ARPA's solution is to the optimal during t, i.e., $\frac{C^t}{C_{OPT}} \rightarrow 1$. Fig. 7 encases $\frac{C^t}{C_{OPT}}$ as a function of time, averaged over 10 independent runs for each different scenario. Note, that the 95% confidence intervals are small and so they are not included in the plots.

Apparently, $\frac{C^t}{C_{OPT}}$ fast converges towards its minimum (i.e., $\frac{C^t}{C_{OPT}} = 1$) offering negligible deviance from the optimal solution for all depicted scenarios. In fact, total cost reduction ranges from 81% to 92% when the ARPA terminates for $t = T$. From Fig. 7(a) we verify that connectivity radius indeed affects the speed of reduction, since higher values (e.g., for $\xi = 0.4$) allow the fog nodes to reach their associated renderers with fewer link hops, yielding smaller-sized $\mathcal{SP}$s and thus decreased render costs. On the other hand, from Fig. 7(b) we confirm that a lower host cost allows for more service replicas in the FAN (e.g., for $C^{host}_s = 0.03$), thus steeper reduction. Finally, the same applies for lower latency thresholds as revealed by Fig. 7(c), since more replications are unavoidable in order to guarantee acceptable render delays, resulting in faster reduction. Nevertheless, independently of the replicas created in the FAN, the migration and consolidation prop-

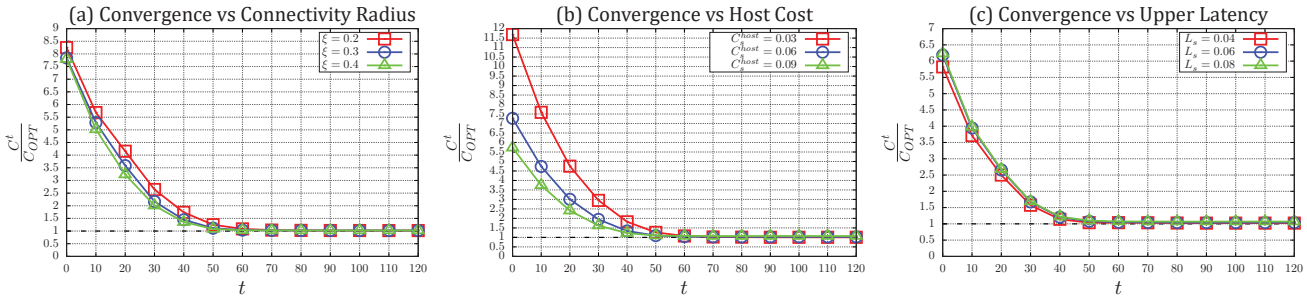

Fig. 7 – $\frac{C^t}{C_{OPT}}$ as a function of t averaged over 10 independent runs, for: (a) various values of ξ with fixed $C_s^{host} = 0.05$ and $L_s = 0.05$; (b) various values of C_s^{host} with fixed $\xi = 0.3$ and $L_s = 0.09$; and (c) various values of L_s with fixed $\xi = 0.3$ and $C_s^{host} = 0.08$.

erties of the ARPA will always shift the placement towards the most favorable locations within the FAN, thus the minimization objective will always converge to a solution closely approximating the optimal one, validating in this way the theoretical analysis.

9. TRACE-DRIVEN EVALUATION

To truly contextualize the efficiency of ARPA, in this section we conduct trace-driven simulations using the OMNeT++ V5.6.2, where we compare our approach to existing alternatives.

9.1 Data set and configuration

For the simulations, the publicly released data set of the EUA Repository[4] is utilized. This particular data set contains geographical coordinates of cellular Base Stations (BSs) in Australia and has been widely utilized in research on EC (e.g.,[66]). Here, Melbourne's "Central Business District" (CBD) covering an area of $6.2\ km^2$ is selected, normalized in a (1×1) plane to form the FAN topology. In CBD there exist a total of 125 Optus BSs, which are accordingly mapped to 125 fog nodes. The main difference between this data set and the simulation configurations in the preceding section lies in the fact that the BSs are now not uniformly distributed across the plane. Instead, there exist areas that are more densely populated as we move towards the busiest parts of the CBD.

The fog nodes form connections based on their BSs' coverage. Different values for ξ are used in the experiments to mirror cellular towers having different communication power. The average rendering demands of each fog node u are randomly selected during initialization from $r^t(u) \in [0, 1], \forall u \in V$ to capture various needs of user proportionality or game graphics qualities within the different BSs' connectivity radii. Again, the topology is considered fixed. For the execution of ARPA, estimations in regards to $R^t(u)$ and $\hat{C}^t_{ren}(u)$ ($\forall u \in V$), take place considering a fixed time slot of $t = 10$ seconds. Likewise, any fog node $v \in V$ may become renderer and so the $\mathcal{SP}$s are once more derived using Dijkstra, according to $D(u, v_s)$ per t.

[4]https://github.com/swinedge/eua-dataset

Each experiment terminates at $t = T = 1000$ seconds. One s is arbitrarily activated on a random fog node during the FAN initialization, defining the first fog renderer. To increase validity, we execute the algorithm ten times and the results are then averaged.

Three sets of experiments are conducted, as follows.

- **Varying Connectivity Radii** (Set #1): Three values are considered; that is, $\xi = \{0.15, 0.25, 0.35\}$. In this way, the topology ranges from marginally to adequately to well-connected.
- **Varying Upper Latency Bound** (Set #2): Three representative values are taken into account; that is, $L_s = \{0.07, 0.1, 0.13\}$. By doing so, views regarding different multimedia delay tolerances are provided.
- **Varying Host Cost** (Set #3): Three values are explored; that is, $C_s^{host} = \{0.04, 0.06, 0.08\}$. As such, insights to diversified host cost-driven placements are exhibited.

For diversity, different seeds are used per experimental set.

9.2 Performance benchmarks

We compare ARPA's performance against five benchmark alternatives, which are modified appropriately to address the LB-UFRL by acknowledging constraint *(4)*, along with the optimal solution under the same conditions:

i) *Greedy* (GR) [29]: In the first iteration of the algorithm, each fog node $u \notin F_s^t$ is individually evaluated to determine their suitability for hosting an s. To this end, the access cost associated with each site under the assumption that the rendering demand flows converge at that site is computed, and eventually the fog node that yields the lowest overall provision cost is picked as renderer. In each following iteration, the next suitable candidate site is searched which, in conjunction with all previously selected ones, results in the lowest overall provision cost under the hypothesis that all fog nodes redirect their rendering demands to the closest available renderer. The algorithm terminates when all fog nodes report a render

cost below the L_s upper latency bound. Note, that in [29], within the context of the web server replica placement problem, it was shown that GR performs close to the optimal solution. Since then it has been widely used in other contexts including edge service provisioning, e.g., [67].

ii) *Random* (RA) [29]: This is a baseline algorithm, wherein each iteration one fog node is randomly selected to become renderer. Termination comes when all fog nodes yield a render cost that does not violate the L_s limit. To increase validity, we execute the algorithm ten times using different seeds and the results are then averaged.

iii) *Hot Spot* (HS) [29]: The algorithm places rendering services by iteratively selecting fog nodes that report the greatest workload in their one-hop neighborhood (derived by ξ). Initially, it sorts all $u \in V$ in descending order according to the traffic generated within their vicinity (including their own rendering demands). It then sequentially selects the top ones until the render cost of all fog nodes drops below the L_s threshold.

iv) *Edge Application Deployment Approximation* (EAD) [54]: The algorithm was recently proposed to address the edge application deployment for the k-facility location problem, under the name EAD-*apx*. For brevity, we omit the {-*apx*} suffix here. EAD runs iteratively. At each round, it sorts all non-renderer nodes in decreasing order of their number of satisfied nodes, i.e., those that obey the L_s bound. Then, it selects the first (top) node to become the next renderer. The algorithm breaks ties according to the maximum delay experienced by the satisfied nodes. The process repeats until the render cost of all fog nodes adheres to the delay constraint.

v) *Betweenness Centrality Depth* (BCD) [55]: It is a recent algorithm where initially a random node is selected as renderer. Then, according to the $\mathcal{SP}$s that are created, the BC degree times the delay to the closest instantiated renderer is computed for all nodes and used to sort the fog nodes in descending order. Then the algorithm selects the first (top) fog node to host the next rendering service. The process repeats until the render cost of all fog nodes abides by the L_s limit. To increase validity, we execute the algorithm ten times and the results are then averaged.

vi) *Optimal* (OPT): This represents the optimal solution to the LB-UFRL problem obtained by the IBM ILOG CPLEX Optimizer solver.

Different seeds are used per run to randomize the initial placement of ARPA, RA and BCD.

9.3 Comparison results

In Fig. 8 we plot the results from Set #1 for fixed parameters relating to host cost and upper latency. Especially regarding the latter, and to increase fairness amongst the different algorithms, we consider a rather high upper latency threshold since the locations of some BSs in the outskirts of CBD, where the network density is rather small, incur unavoidable communication delays due to long distances amongst the participating fog nodes.

We observe from Fig. 8(a) that the BCD yields in all cases the smallest $|F_s^T|$ since it fast determines hub nodes that are jointly critical to the created $\mathcal{SP}$s and overall render cost delays of the served nodes. BCD is followed by GR which employs exhaustive searches to tackle the delay constraint. EAD is also very effective in this matter. In fact, as connectivity radius becomes larger, EAD's location output becomes slightly smaller since its primary objective is the minimization of the number of instantiated rendering services, a fact that is assisted by the increase in ξ due to more nodes being satisfied by fewer renderers. ARPA, on the other hand, yields in all cases the fourth lowest placement (i.e., 42 renderers on average) which, however, is observed to actually be the closest to OPT (i.e., 43 renderers here). Because the optimal $\mathcal{SP}$s in CBD are not altered by the particular ξ values, we notice that OPT remains the same across all runs. RA is significantly worse (with a margin of at least 40.33% higher placement than ARPA) which is understandable since it remains oblivious to any network information other than the alleviation of L_s. Interestingly, HS performs the worst, resulting in a high number of activated renderers (e.g., for $\xi = 0.35$ HS reaches the extreme value of $|F_s^T| = 104$). The behavior of HS is attributed to the non-uniform positioning of the BSs in CBD which in turn allows the denser areas to host the highest workload within the nodes' neighborhoods, rendering more complex the enforcement of the L_s bound to all other nodes in less populated areas. The above properties are perfectly mirrored in Fig. 8(b), where we verify that ARPA indeed best approximates OPT's C_{depl}^T out of all alternatives.

The efficiency of ARPA, nevertheless, is more accurately depicted in Fig. 8(c) where once more it is revealed that it performs the closest to OPT in terms of C_{prov}^T, especially as ξ increases. In contrast, GR here exhibits a far worse behavior as ξ expands, yielding more than double provision cost (122.31% increment over ARPA when $\xi = 0.25$ and 118.56% when $\xi = 0.35$) . RA's performance is heavily affected by its final placement and so remains independent of ξ, whereas HS leads to increasingly better re- sults as the connectivity radius rises, since the workload also increases at each neighborhood, yielding for $\xi = 0.35$ the lowest C_{prov}^T (but as already stated the largest place- ment). EAD also performs quite poorly, with a tendency to offer slightly better provision as ξ expands and more effi- cient location of the renderers are discovered. Lastly, BCD presents the worst behavior in terms of provision cost which is expected when in

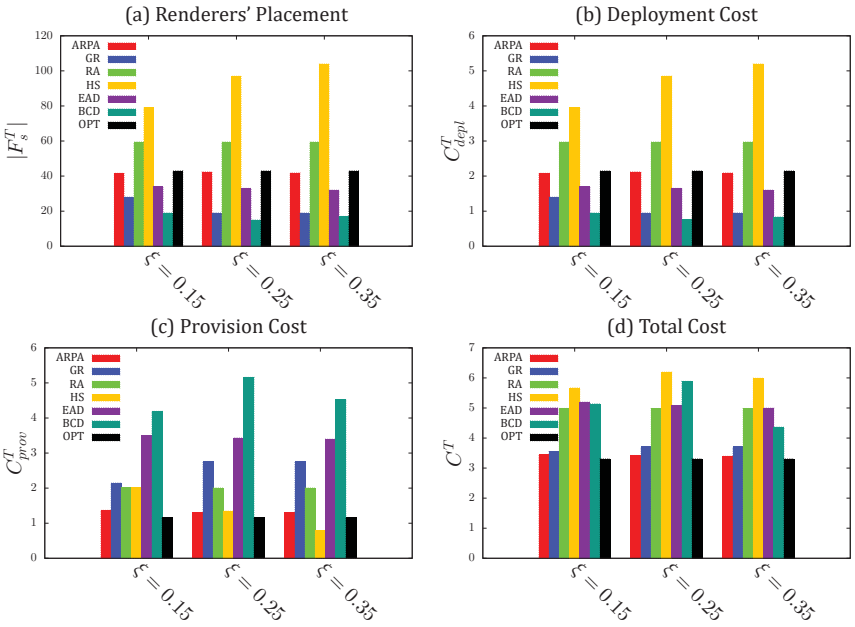

Fig. 8 – Comparisons with fixed $C_s^{host} = 0.05$ and $L_s = 0.12$, for three values of the connectivity radius ξ.

conjunction with the smallest resulted placement.

When summing up C_{prov}^T and C_{depl}^T for each algorithm, we remark from Fig. 8(d) that ARPA best approximates OPT (with a gap of $\simeq 4\%$ that further closes as ξ expands) exceedingly outperforming all other algorithms, even the purely centralized GR. In fact, GR performs slightly worse with an inverted attitude, yielding slowly increasing C^T when ξ rises reaching a margin of 12.16% over OPT. HS is the worst with a far greater total cost in all cases. Depending on ξ, RA, EAD and BCD also perform rather poorly and significantly worse than GR and thus ARPA. These findings are in compliance with the theoretical analysis and verify the efficiency of ARPA in optimizing the trade-off between both deployment and provision costs, closely converging to the optimal placement solution OPT.

Following the previous observations, in Fig. 9 we encapsulate results for Set #2, where we compare the various algorithms under different multimedia delay tolerances while fixing the host cost and connectivity radius of the BSs. Notice once more that OPT remains here practically unaffected by the different upper latency cases since for the highest value, i.e., when $L_s = 0.13$, it reaches an optimal placement that can also satisfy the lowest value, i.e., when $L_s = 0.07$. The cost distribution of most algorithms (except ARPA) follows in general the same principles with some small differences relating to L_s. These are as follows.

From Fig. 9(a) it is obvious that ARPA presents a rather stable behavior, managing to sustain the number of renderers at relatively low levels independently of L_s, yielding the closest (i.e., 31 renderers on average) to OPT (i.e., 32 renderers) location. All other algorithms follow the same pattern, increasingly instantiating fewer renderers as the L_s bound is relaxed. Evidently, BCD produces repeatedly the smallest $|F_s^T|$ in all cases, a clear testament to its ability to weigh the popularity of each node against the incurred delay. For GR, on the other hand, the turning point is the case where $L_s = 0.1$ since for lesser values it locates substantially more renderers (31.41%) than ARPA, whereas for higher values it yields significantly less (58.59%), while for $L_s = 0.1$ their performance is close in terms of deployment cost. Apparently, RA and EAD also behave similarly, with EAD having a significantly higher dropping rate as delay tolerance increases from $L_s = 0.1$ to $L_s = 0.13$. Like in Set #1, HS yields inferior results, activating far more renderers since it neglects fog nodes with a lower neighborhood workload while prioritizing hot spots where fog nodes are heavily packed near each other. Nonetheless, it also exhibits a fast decay of renderers as L_s rises, in which case the communication delay in Eq. *(3)* fast compensates for the increased workload, leading to higher rates of satisfied nodes adhering to the delay threshold. The aforementioned are again perfectly mapped to the overall deployment cost captured in Fig. 9(b), where it is obvious that ARPA achieves the greatest approximation to the OPT deployment solution.

Regarding the overall provision cost, highlighted in Fig. 9(c), we can observe that GR, RA, and EAD for $L_s = 0.07$ yield improved results compared to ARPA (due to the higher deployment cost) and close to OPT, with the first (i.e., GR) being the most efficient. However, as L_s gets relaxed, the placement properties of ARPA allow the renderers to more freely and accurately relocate to favorable positions, without violating the upper latency bound. Hence, for higher values, ARPA begins to narrow the initial performance gap and output increasingly better provision cost, whereas the rest follow a converse attitude, steadily offering worse results. Interestingly, even GR with an average $>30\%$ of C_{prov}^T growth among the consecutive L_s values that are tested, is outperformed by ARPA

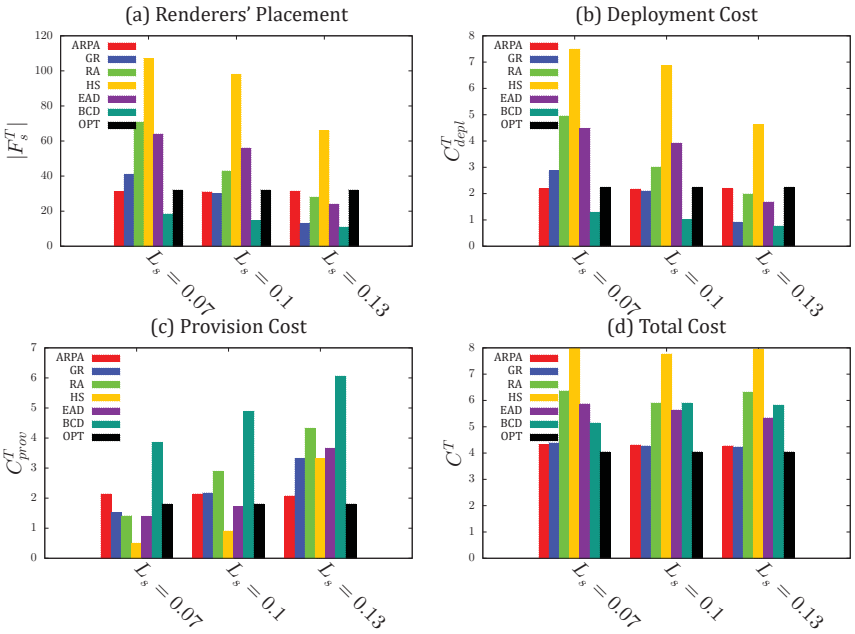

Fig. 9 – Comparisons with fixed $C_s^{host} = 0.07$ and $\xi = 0.2$, for three values of the upper latency bound L_s.

for $L_s = \{0.1, 0.13\}$, resulting in a margin of $\simeq 2\%$ and 37.52% over ARPA, respectively. HS presets the highest factor of increment which is coupled with the greatest reduction rate in the number of renderers, as stated previously. RA and EAD also yield gradually higher C_{prov}^T with the former having a substantially larger percentage of growth. For the latter, it is noteworthy that for $L_s = 0.1$ it manages to approximate OPT better than any alternative, which however is in accordance with the second largest placement as observed previously. Yet, L_s again has a negative impact on EAD for higher values because, as it expands, EAD's properties allow hub renderers (i.e., those that are closest to the center of the denser network sites) to progressively satisfy more distant fog nodes, which however are served now with an analogous increment in their render cost. BCD, as expected, generates the highest provision cost in all cases which additionally worsens as multimedia latency tolerance becomes greater, a fact that is perfectly coupled with its ability to produce steadily smaller $|F_s^T|$. This indicates its suitability for minimizing the deployment cost, which however is not weighed against the loss in service provisioning.

Summarizing both deployment and provision costs, in Fig. 9(d) we plot the total cost acquired by each algorithm. Clearly, GR and ARPA outperform the alternatives by a great magnitude (with a solution at most 9% away from OPT), with the former yielding slightly better results (a performance gap of less than 2% on average from ARPA). Still, different than GR which is a purely centralized algorithm that employs global knowledge to approximate OPT, ARPA is a completely decentralized solution. Therefore, ARPA's high approximation ability can be exceedingly leveraged in dynamic, mobile, or large-scale multimedia systems where changes in network conditions frequently occur, and so continuous exhaustive searches, such as the ones executed by GR, are not viable in the long run in terms of computational complexity and cost. EAD, following an overall affinity to reduce the C^T as L_s grows, seems to be the next best option when considering that HS repeatedly fails to minimize the total cost, while BCD presents the opposite trend with a tendency to increase C^T (note, here, that lower L_s values favor BCD). Finally, RA remains impartial offering poor performance under any L_s. The aforementioned highlight the superiority of ARPA in addressing the objective function of LB-UFRL, i.e., the trade-off between both the gain and the loss of the overall rendering service deployment and provision costs, and in the meanwhile provide strict multimedia delay guarantees that are obeyed under all circumstances to offer maximum QoS.

In regards to Set #3 and to increase fairness, we only provide insights regarding the behavior of ARPA compared to OPT and omit all other algorithms since they do not factor in during their execution the value of C_s^{host}. Fig. 10 encases the obtained results under fixed $\xi = 0.2$ and $L_s = 0.08$. The particular values are intentionally set to low in order to observe how ARPA reacts and adjusts the placement after the delay has been compensated for weakly-connected networks. Apparently, from Fig. 10(a) and 10(b) we can remark that ARPA achieves a good approximation of OPT in terms of C_{depl}^T, managing a deviation of 11.17% from OPT when $C_s^{host} = 0.08$, while for lesser values, i.e., when $C_s^{host} = 0.06$ and $C_s^{host} = 0.04$, its performance gap is reduced to 6.34% and 3.12% respectively, since more replications take place and service consolidation can transpire more easily. The opposite trends are revealed by Fig. 10(c) where the margin for C_{prov}^T between the two algorithms tends to increase, which is actually to be expected when in combination with the previous

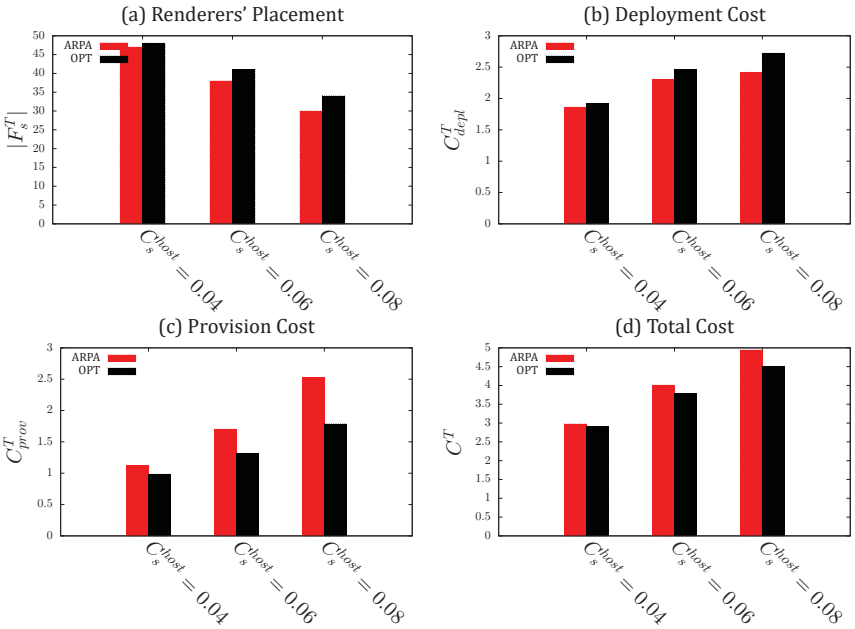

Fig. 10 – Comparisons against the optimal solution with fixed $L_s = 0.08$ and $\xi = 0.2$, for three values of the host cost C_s^{host}.

findings and the accelerated reduction rate in the renderers (i.e., as host cost grows) in which case the provision gain from deploying more rendering services becomes cost-unworthy for the provider. Despite the negative impact on service provisioning, the total cost posed by ARPA remains relatively close to OPT, as reported by Fig. 10(d). In fact, even for the worst case (i.e, for $C_s^{host} = 0.08$) ARPA still manages to sufficiently converge towards the optimal solution with a divergence of 9.64% from OPT, while for a lesser host cost, i.e., $C_s^{host} = 0.06$ and $C_s^{host} = 0.04$, the performance gap reduces to 6.13% and 2.55% accordingly, indicating that the prevention rate of ARPA's relocation properties in respect to replication and consolidation decreases significantly, allowing for an augmentation in the final placement.

The last claim, along with all previous observations made, are validated further by Table 2, which provides the average number of occurrences for each relocation rule (i.e., migration, replication, and consolidation) for all conducted experimental sets and scenarios. For Set #1 it is obvious that higher values of ξ favor fewer relocations since the rise in connectivity radius allows for faster convergence. Noteworthy is the fact that for $\xi = 0.15$, the migration frequency more than triples that compared to $\xi = 0.35$ since, as already mentioned, the number of links is significantly smaller which in turn necessitates more service movements to lead the renderers to suitable locations within the FAN. For the same reason, consolidation is commended for lower values of ξ, because longer distances render inefficient the operation of extra renderers (i.e., Rule 4). For Set #2 we can remark that as the multimedia delay tolerance increases, fewer replications are required to compensate for L_s (i.e., Rule 2). As a result of the lesser number of replicas present, the rate of consolidation also drops, hence, migration takes over to optimize the placement. Finally, regarding Set #3 we observe that a higher host cost indeed halts the replication process significantly (i.e., Rule 3). As such, consolidation and migration also experience a decay (since fewer replica candidates satisfy ARPA's relocation Rules 1 and 4), a fact that explains the growth in the overall service provision cost.

Table 2 – ARPA's average number of rendering service relocations per experimental set and scenario.

Set	Scenario	Migrations	Replications	Consolidations
#1	$\xi = 0.15$	20.6	43.6	3.8
	$\xi = 0.25$	8.8	42.4	2.9
	$\xi = 0.35$	6.1	41.8	1.9
#2	$L_s = 0.07$	9.6	32.5	3.1
	$L_s = 0.10$	9.7	31.7	2.5
	$L_s = 0.13$	10	31.5	2.3
#3	$C_s^{host} = 0.04$	10.8	49.5	4.9
	$C_s^{host} = 0.06$	9.3	38.3	1.8
	$C_s^{host} = 0.08$	8.2	29.8	1.5

10. CONCLUSION

In this paper, the problem of efficient placement of renderers within fog-assisted cloud multimedia computing systems was addressed and studied. Based on the cloud/fog architecture, the goal was to locate rendering services at key locations within the fog layer, in order to achieve timely service provision with simultaneous deployment cost minimization, meeting stringent delay guarantees regarding the expected 3D rendering time.

To address the particular problem, a constrained optimization approach was followed to mathematically formulate the problem within a finite time horizon. Given its high complexity, a thorough theoretical analysis was provided that designates the minimum acceptable conditions for relocating the services towards optimal locations, based solely on localized network information, that

iteratively lead to total cost reduction during each discrete time slot until a solution is found. Subsequently, a distributed algorithm, namely the ARPA, capitalizing on the analysis was proposed and discussed. The properties of ARPA allow the renderers to autonomously migrate, replicate, or complementarily merge, until an optimal placement has been reached.

Comprehensive simulations under various deployment scenarios, in accordance with the analytical findings, confirmed the algorithm's expected behavior and proved its efficiency for fast convergence and delay guarantees, even when fluctuations appear regarding the network conditions during runtime. As such, ARPA was shown to be resilient to changes and capable of scaling up and down to meet the current demands, producing placement solutions that closely approximate the optimal one. Through additional trace-driven evaluations, we also showcased ARPA's robustness and applicability on realistic topologies. The results clearly exhibited its efficacy in outperforming past approaches and in generating near-optimal placements that rival even the solutions of the purely centralized greedy approach.

Future research will involve ARPA's real-world deployment using appropriate test beds. Besides, no relocation cost was assumed here during the placement, and no capacities were reported in the model. Thus, in future work, the impact of such dependencies will also be investigated.

A. APPENDIX

Here you may find the proofs of the various lemmas and theorems in their order of appearance within the paper.

A.1 Proof of Lemma 1

When migration takes place at a given time slot, the renderers' placement changes for the next time slot. Therefore, there exists a cost difference between the two consecutive time slots attributed to the new $\mathcal{SP}$s that are formed. However, to calculate this difference would require global topology knowledge. Thus, assuming that ${}^t\overrightarrow{x_s : y}$ occurs during the time slot t and the service s has moved *instantaneously* to the new location within $\delta(x_s) \cap S^t_{x_s}$, then there exists a new *M-hypothetical total cost*, denoted as $\overrightarrow{C^t}$, incurred by the new *M-hypothetical placement*, denoted as $\overrightarrow{F^t_s}$, after migration. Then, the render cost $\forall u \in S^t_{x_s}$ is also affected when taking into account the fact that their $r^t(u)$ are accumulated along the established $\mathcal{SP}$ and up to the previous x_s. On the contrary, the remainder fog nodes of the FAN (i.e., $F^t_s \backslash S^t_{x_s}$) remain practically unaffected by this change, since they are served by different renderers that do not contribute additional cost differences to the $\overrightarrow{C^t}$ when their $\mathcal{SP}$s remain unaltered during the instantaneous migration. Let $y \in \delta(x_s) \cap S^t_{x_s}$ be the neighbor node where $d(x_s, y) > d(x_s, z), \forall z \in \delta(x_s) \cap S^t_{x_s}$ (i.e., the worst-case scenario). Consequently, the minimum post-migration cost difference can now be expressed as $\Delta C({}^t\overrightarrow{x_s : y})_{min} = C^t - C^{t+1} = C^t - \overrightarrow{C^t} = C^t_{prov} + C^t_{depl} - \overrightarrow{C^t_{prov}} - \overrightarrow{C^t_{depl}}$. However, since $|F^t_s| = |\overrightarrow{F^t_s}|$ then $C^t_{depl} = \overrightarrow{C^t_{depl}}$ and the previous statement can now be written as $\Delta C({}^t\overrightarrow{x_s : y})_{min} = \sum_{\forall u \in S^t_{x_s}} C^t_{ren}(x_{s \to u}) - \sum_{\forall u \in S^t_{y_s}} \overrightarrow{C^t_{ren}}(y_{s \to u})$. However, from Eq. *(3)* we have $\Delta C({}^t\overrightarrow{x_s : y})_{min} = \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) - \sum_{\forall u \in S^t_{y_s}} r^t(u) D(u, y_s)$, or equivalently $\Delta C({}^t\overrightarrow{x_s : y})_{min} = C^t_{acc}(x_s) - \overrightarrow{C^t_{acc}}(y_s)$, and eventually $\Delta C({}^t\overrightarrow{x_s : y})_{min} = C^t_{acc}(x_s) - C^t_{acc}(y)$.

A.2 Proof of Theorem 1

Based on Lemma 1 we have $\Delta C({}^t\overrightarrow{x_s : y})_{min} = C^t_{acc}(x_s) - C^t_{acc}(y)$. To determine if $\Delta C({}^t\overrightarrow{x_s : y}) > 0, \forall y \in \delta(x_s) \cap S^t_{x_s}$, and in turn if $C^t > C^{t+1}$, it is sufficient enough to show when $\Delta C({}^t\overrightarrow{x_s : y})_{min} > 0 \Leftrightarrow C^t_{acc}(x_s) - C^t_{acc}(y) > 0 \Leftrightarrow C^t_{acc}(x_s) > C^t_{acc}(y) \Leftrightarrow C^t_{acc}(x_s) > \overrightarrow{C^t_{acc}}(y_s)$. Note that $C^t_{acc}(x_s) = \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) = \sum_{\forall u \in S^t_{x_s} \backslash \check{S}^t_{x_s:y}} r^t(u) D(u, x_s) + \sum_{\forall u \in \check{S}^t_{x_s:y}} r^t(u) D(u, y) + R^t(y) d(y, x_s)$. Likewise, $\overrightarrow{C^t_{acc}}(y_s) = \sum_{\forall u \in S^t_{y_s}} r^t(u) D(u, y_s) = \sum_{\forall u \in S^t_{y_s} \backslash \check{S}^t_{y_s:x}} r^t(u) D(u, y_s) + \sum_{\forall u \in \check{S}^t_{y_s:x}} r^t(u) D(u, x) + \overrightarrow{R^t}(x) d(x, y_s)$, where $\overrightarrow{R^t}(x)$ are the *M-hypothetical aggregate rendering demands* of fog node x after the migration. By subtracting $\overrightarrow{C^t_{acc}}(y_s)$ from $C^t_{acc}(x_s)$ we get $C^t_{acc}(x_s) - \overrightarrow{C^t_{acc}}(y_s) = \sum_{\forall u \in S^t_{x_s} \backslash \check{S}^t_{x_s:y}} r^t(u) D(u, x_s) + \sum_{\forall u \in \check{S}^t_{x_s:y}} r^t(u) D(u, y) + R^t(y) d(y, x_s) - \sum_{\forall u \in S^t_{y_s} \backslash \check{S}^t_{y_s:x}} r^t(u) D(u, y_s) - \sum_{\forall u \in \check{S}^t_{y_s:x}} r^t(u) D(u, x) - \overrightarrow{R^t}(x) d(x, y_s)$. Because $d(y, x_s) = d(x, y_s) = d(x, y)$, and since $S^t_{x_s} \backslash \check{S}^t_{x_s:y} = \check{S}^t_{y_s:x}$ and $\check{S}^t_{x_s:y} = S^t_{y_s} \backslash \check{S}^t_{y_s:x}$ (otherwise the nodes will have preferred a closer renderer with less delay which definitely leads to total cost decay), then $C^t_{acc}(x_s) - \overrightarrow{C^t_{acc}}(y_s) = R^t(y) d(x, y) - \overrightarrow{R^t}(x) d(x, y) = (R^t(y) - \overrightarrow{R^t}(x)) d(x, y)$. Therefore, we require $\Delta C({}^t\overrightarrow{x_s : y})_{min} = (R^t(y) - \overrightarrow{R^t}(x)) d(x, y) > 0$. Since by default $d(x, y) > 0$ then for $\Delta C({}^t\overrightarrow{x_s : y})_{min} > 0$ it is enough for $R^t(y) - \overrightarrow{R^t}(x) > 0$. However, $\overrightarrow{R^t}(x) = R^t(x_s) - R^t(y)$, and so $R^t(y) - \overrightarrow{R^t}(x) = R^t(y) - (R^t(x_s) - R^t(y)) = R^t(y) - R^t(x_s) + R^t(y) > 0 \Leftrightarrow \Delta C({}^t\overrightarrow{x_s : y})_{min} = 2R^t(y) - R^t(x_s) > 0$. This is a lower bound condition for the minimum post-migration cost difference in order to reach total cost reduction, and so considering the general case, it can be inferred that for any $y \in \delta(x_s) \cap S^t_{x_s}$ if $\Delta C({}^t\overrightarrow{x_s : y}) \geq 2R^t(y) - R^t(x_s) > 0$ then $C^t - C^{t+1} > 0$ is also satisfied.

A.3 Proof of Lemma 2

Since service replication ${}^t\overleftarrow{x_s : y'}$ will create a new renderer at the fog node $y' \in V$, then it is ensured that $F^{t+1}_s = F^t_s \cup \{y'_s\}$ and so $C^t_{depl} < C^{t+1}_{depl}$. Obviously,

fog nodes where $D(u, y'_s) < D(u, x_s), \forall u \in V, \forall x_s \in F_s^{t+1} \backslash y'_s$ will prefer to be served by the former renderer, i.e., y'_s. Thus, it is trivial to conclude that in this case $C^t_{ren}(x_{s\to u}) > C^{t+1}_{ren}(y'_{s\to u})$, or $C^t_{acc}(x_s) > C^{t+1}_{acc}(y'_s)$, and in turn $C^t_{prov} > C^{t+1}_{prov}$.

A.4 Proof of Theorem 2

First, when ${}^t\overrightarrow{x_s : y'}$ then it is ensured that $F_s^{t+1} = F_s^t \cup \{y'_s\}$, and thus $C^t = C^{t+1} + \Delta C({}^t\overrightarrow{x_s : y'})$. Second, in order for $C^t > C^{t+1}$ we require that $\Delta C({}^t\overrightarrow{x_s : y'}) > 0$. Still, in view of Lemma 2 it is impossible to know beforehand how C^{t+1} will be affected by the $\mathcal{SP}$ alterations caused by the replication, since this would require global topology knowledge. Instead, assuming that ${}^t\overrightarrow{x_s : y'}$ happens *instantaneously* during t and no $\mathcal{SP}$ recreations occur that will surely increase the $\Delta C({}^t\overrightarrow{x_s : y'})$, then there is an *R-hypothetical total cost*, denoted as $\overrightarrow{C^t}$, incurred by the new *R-hypothetical placement*, denoted as $\overrightarrow{F_s^t}$, after replication. Then, the post-replication cost difference becomes $\Delta C({}^t\overrightarrow{x_s : y'}) = C^t - \overrightarrow{C^t}$. Note that via Eq. *(8)* the $C^t = C^t_{prov} + C^t_{depl} = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S^t_{v_s}} C^t_{ren}(v_{s\to u}) + \sum_{\forall v_s \in F_s^t} C^{host}_s = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t| C^{host}_s$. Because our primary interest is on the particular renderer that participates in the replication process, then this relation is equivalent to writing $C^t = \sum_{\forall v_s \in F_s^t \backslash \{x_s\}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t \backslash \{x_s\}| C^{host}_s + \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) + C^{host}_s$. Likewise, $\overrightarrow{C^t} = \overrightarrow{C^t_{prov}} + \overrightarrow{C^t_{depl}} = \sum_{\forall v_s \in \overrightarrow{F_s^t}} \sum_{\forall u \in S^t_{v_s}} \overrightarrow{C^t_{ren}}(v_{s\to u}) + \sum_{\forall v_s \in \overrightarrow{F_s^t}} C^{host}_s = \sum_{\forall v_s \in \overrightarrow{F_s^t}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |\overrightarrow{F_s^t}| C^{host}_s$. However, considering that $\overrightarrow{F_s^t} = F_s^t \cup \{y'\}$, then $\overrightarrow{C^t} = \sum_{\forall v_s \in F_s^t \backslash \{x_s\}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t \backslash \{x_s\}| C^{host}_s + \sum_{\forall u \in \overrightarrow{S^t_{x_s}}} r^t(u) D(u, x_s) + C^{host}_s + \sum_{\forall u \in \overrightarrow{S^t_{y'_s}}} r^t(u) D(u, y'_s) + C^{host}_s$. Thus, by subtracting the two costs we quickly arrive to the result $\Delta C({}^t\overrightarrow{x_s : y'}) = \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) - \sum_{\forall u \in \overrightarrow{S^t_{x_s}}} r^t(u) D(u, x_s) - \sum_{\forall u \in \overrightarrow{S^t_{y'_s}}} r^t(u) D(u, y'_s) - C^{host}_s \Leftrightarrow \Delta C({}^t\overrightarrow{x_s : y'}) = \sum_{\forall u \in S^t_{x_s} \backslash \check{S}^t_{x_s : y'}} r^t(u) D(u, x_s) + \sum_{\forall u \in \check{S}^t_{x_s : y'}} r^t(u) D(u, y') + R^t(y') d(y', x_s) - \sum_{\forall u \in \overrightarrow{S^t_{x_s}}} r^t(u) D(u, x_s) - \sum_{\forall u \in \overrightarrow{S^t_{y'_s}}} r^t(u) D(u, y'_s) - C^{host}_s$. Since $S^t_{x_s} \backslash \check{S}^t_{x_s : y'} = \overrightarrow{S^t_{x_s}}$ and $\check{S}^t_{x_s : y'} = \overrightarrow{S^t_{y'_s}}$, i.e., they are identical, then ultimately $\Delta C({}^t\overrightarrow{x_s : y'}) = R^t(y') d(y', x_s) - C^{host}_s$. Subsequently, for $C^t > \overrightarrow{C^t}$ it is enough $\Delta C({}^t\overrightarrow{x_s : y'})_{min} = \{\Delta C({}^t\overrightarrow{x_s : y'}) : R^t(y') d(y', x_s) \leq R^t(z) d(z, x_s), \forall z \in \delta(x_s) \cap S^t_{x_s}\} > 0$ (i.e., the worst-case scenario). The last is a lower bound condition for the minimum post-replication cost difference in order to achieve total cost reduction, and so considering the general case, for any $y' \in \delta(x_s) \cap S^t_{x_s}$, if $\Delta C({}^t\overrightarrow{x_s : y'}) \geq R^t(y') d(y', x_s) - C^{host}_s > 0$ then $C^t - C^{t+1} > 0$ is also satisfied.

A.5 Proof of Lemma 3

Since service consolidation ${}^t\overleftarrow{x_s : y''_s}$ will merge two renderers, then it is ensured that $F_s^{t+1} = F_s^t \backslash \{y''_s\}$ and so $C^t_{depl} > C^{t+1}_{depl}$. Then, fog nodes where $D(u, y''_s) < D(u, x_s), \forall u \in V, \forall x_s \in F_s^t$ will for $t = t + 1$ be served by a new renderer, i.e., some $x_s \in F_s^{t+1}$. Thus, it is trivial to conclude that in this case $C^{t+1}_{ren}(x_{s\to u}) > C^t_{ren}(y''_{s\to u})$, or $C^t_{acc}(x_s) > C^{t+1}_{acc}(y'_s)$, and in turn $C^t_{prov} < C^{t+1}_{prov}$.

A.6 Proof of Theorem 3

The proof is similar to that of Theorem 2. When ${}^t\overleftarrow{x_s : y''_s}$ then it is ensured that $F_s^{t+1} = F_s^t \backslash \{y''_s\}$, and thus $C^t = C^{t+1} + \Delta C({}^t\overleftarrow{x_s : y''_s})$. Second, in order for $C^t > C^{t+1}$ we necessitate that $\Delta C({}^t\overleftarrow{x_s : y''_s}) > 0$. Yet, in view of Lemma 3 it is impossible to predict how C^{t+1} will be affected since this would demand global knowledge. Instead, assuming that ${}^t\overleftarrow{x_s : y''_s}$ happens *instantaneously* during t and no $\mathcal{SP}$ reformations happen, then there is a *C-hypothetical total cost*, denoted as $\overleftarrow{C^t}$, due to the new *C-hypothetical placement*, denoted as $\overleftarrow{F_s^t}$, after consolidation takes place. Like migration, we also demand here that $d(y''_s, x_s) > d(y''_s, v_s), \forall v_s \in F_s^t \cap \delta(y''_s)$ (i.e., the worst-case scenario), otherwise the rendering demands of all $u \in S^t_{y''_s}$ will for $t = t + 1$ be forwarded to other renderers with less rendering costs, which in turn will further increase $\Delta C({}^t\overleftarrow{x_s : y''_s})$. Thus, to obtain the lower bound we must enforce that x_s is the worst candidate for the consolidation. Then, the minimum post-consolidation cost difference becomes $\Delta C({}^t\overleftarrow{x_s : y''_s})_{min} = C^t - \overleftarrow{C^t}$. Through Eq. *(8)* the $C^t = C^t_{prov} + C^t_{depl} = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S^t_{v_s}} C^t_{ren}(v_{s\to u}) + \sum_{\forall v_s \in F_s^t} C^{host}_s = \sum_{\forall v_s \in F_s^t} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t| C^{host}_s = \sum_{\forall v_s \in F_s^t \backslash \{x_s, y''_s\}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t \backslash \{x_s, y''_s\}| C^{host}_s + \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) + \sum_{\forall u \in S^t_{y''_s}} r^t(u) D(u, y''_s) + 2C^{host}_s$. Likewise, $\overleftarrow{C^t} = \overleftarrow{C^t_{prov}} + \overleftarrow{C^t_{depl}} = \sum_{\forall v_s \in \overleftarrow{F_s^t}} \sum_{\forall u \in S^t_{v_s}} \overleftarrow{C^t_{ren}}(v_{s\to u}) + \sum_{\forall v_s \in \overleftarrow{F_s^t}} C^{host}_s = \sum_{\forall v_s \in \overleftarrow{F_s^t}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |\overleftarrow{F_s^t}| C^{host}_s$. However, since $\overleftarrow{F_s^t} = F_s^t \backslash \{y''_s\}$, then $\overleftarrow{C^t} = \sum_{\forall v_s \in F_s^t \backslash \{x_s, y''_s\}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t \backslash \{x_s, y''_s\}| C^{host}_s + \sum_{\forall u \in \overleftarrow{S^t_{x_s}}} r^t(u) D(u, x_s) + C^{host}_s = \sum_{\forall v_s \in F_s^t \backslash \{x_s, y''_s\}} \sum_{\forall u \in S^t_{v_s}} r^t(u) D(u, v_s) + |F_s^t \backslash \{x_s, y''_s\}| C^{host}_s + \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) + \sum_{\forall u \in S^t_{y''_s}} r^t(u) D(u, y''_s) + R^t(y''_s) d(y''_s, x_s) + C^{host}_s$. Thus, by subtracting the two costs, $\Delta C({}^t\overleftarrow{x_s : y''_s})_{min} = \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) + \sum_{\forall u \in S^t_{y''_s}} r^t(u) D(u, y''_s) + 2C^{host}_s - \sum_{\forall u \in S^t_{x_s}} r^t(u) D(u, x_s) - \sum_{\forall u \in S^t_{y''_s}} r^t(u) D(u, y''_s) - R^t(y''_s) d(y''_s, x_s) - C^{host}_s \Leftrightarrow$

$\Delta C(^t\overleftarrow{x_s : y_s''})_{min} = C_s^{host} - R^t(y_s'')d(y_s'', x_s) > 0$. This is a lower bound condition for the minimum post-consolidation cost difference to achieve total cost reduction, and so regarding the general case $\forall y_s'' \in \delta(x_s)$ it is proved that if $\Delta C(^t\overleftarrow{x_s : y_s''}) \geq C_s^{host} - R^t(y_s'')d(y_s'', x_s) > 0$ then $C^t - C^{t+1} > 0$ is also satisfied.

A.7 Proof of Lemma 4

Our focus is on the worst-case scenario where the underlying $\mathcal{SP}$s are not affected by the replications. Let $u \in S_{v_s}^t$ represent a fog node where $C_{ren}^t(v_{s\to u}) > L_s$ and let $\hat{C}_{ren}^t(v_s) = \max\{\hat{C}_{ren}^t(\nu)_{\forall \nu \in S_{v_s}^t \cap \delta(v_s)}\} = C_{ren}^t(v_{s\to u})$. Obviously, when $^t\overleftarrow{v_s : y'}$ occurs, then for $t = t+1$ the $C_{acc}^t(x_s) > C_{acc}^{t+1}(\nu_s) \Leftrightarrow C_{ren}^t(v_{s\to u}) > C_{ren}^t(\nu_{s\to u})$. By iteratively replicating services down the $\mathcal{SP}$ connecting the initial v_s to u the render cost of u will continue to drop. This process will repeat until the render cost becomes less than L_s, or in the extreme occasion when u itself becomes a renderer in which case $C_{ren}^{t+k_{u,v_s}}(u_{s\to u}) = r^t(u)D(u, u_s) = r^t(u)d(u, u)$. Since $d(u, u) = 0$, then $C_{ren}^{t+k_{u,v_s}}(u_{s\to u}) = 0$. Thus, it will require at the most k_{u,v_s} replications for $C_{ren}^t(v_{s\to u}) < L_s$ to be satisfied.

A.8 Proof of Theorem 4

This is easily proved by contradiction. Assume that for some arbitrary t, the placement process has been terminated and some node z still exists in the FAN where $C_{ren}^t(\omega_{s\to z}) > L_s$ ($\omega_s \in F_s^t$ and $k_{z,\omega_s} \geq 1$). Then, $\hat{C}_{ren}^t(\omega_s) > L_s$ is also satisfied. Thus, renderer ω_s will immediately initiate a replication process to compensate for L_s. This leads to a contradiction since it is implied that the placement process has not yet ended. Actually, according to Lemma 4 and given that (i) each fog node is served by exactly one renderer (i.e., the one with the minimum render cost), and (ii) no other alterations will appear in the underlying $\mathcal{SP}$s, then it will additionally require at most $t = t + k_{z,\omega_s}$ time slots for the relocation process to terminate. Thus, at the end of the rendering service relocation, no fog node exists that violates L_s.

A.9 Proof of Lemma 5

ARPA's execution is iterative; that is, each renderer per t will attempt to relocate its service according to one (and exactly one) of the rules provided, by sequentially validating their conditions. In the worst case this means traversing though all neighbor nodes of a fog renderer with maximum neighborhood size. Thus, the time complexity is bounded, i.e., $\mathcal{O}(|F_s^t|N)$, where $N \triangleq \max_{\forall u \in V} |\delta(u)|$.

A.10 Proof of Theorem 5

It is suffice to show that there cannot exist loops, i.e., repeated visits to the same configuration of renderers within the FAN. We distinguish four cases depending on the L_s and C_s^{host} that capture all the different relocation alternatives.

α. The initial placement of renderers, i.e., the F_s^0 abides by the L_s constraint. Then, the LB-UFRL effectively reduces to a pure UFL problem where the services relocate in a loop-free manner amongst the fog nodes of G. This means that the placement, following a monotonically decreasing total cost path based on Rules 1, 3 and 4, will visit each combination of potential renderers only once. Since the solution space is finite, the ARPA will terminate after a finite number of time slots, i.e., when the $C^t > C^{t+1}$ criterion ceases to apply or when the ARPA reaches the optimal solution of the LB-UFRL.

β. The initial placement of renderers does not abide by the L_s constraint. Then after the compensation of the L_s is completed via Rule 2, the ARPA will adopt the same behavior as described previously and ultimately converge to a solution where $C^t \leq C^{t+1}$ for all other relocation rules, in which case it will also terminate.

γ. The C_s^{host} is set to a very high value prohibiting the opening of new replicas, i.e., the execution of Rule 3. Then we can further observe two cases. First, following the L_s compensation we will have k renderers placed in the FAN. Thus, the LB-UFRL effectively reduces to a pure k-median problem and the ARPA will reach in a finite number of time slots through Rules 1 and 4 to a solution with up to k renderers present in the FAN. Second, if the initial placement consists of only one renderer, i.e., $|F_s^0| = 1$ and the L_s bound is not infringed, i.e., no other replications or consolidations can occur, the LB-UFRL is transformed to a 1-median problem. Even in this case, the migration criterion in Rule 1 will lead the ARPA to a solution, given the total cost being monotonically decreasing between successive time slots. Thus, the ARPA will again terminate.

δ. The C_s^{host} is set to a very low value which prohibits the merging of services. Once more, after replicas are created (i.e., due to Rules 2 or 3), then Rule 1, because it is executed first, will always migrate the services towards optimized locations that reduce the total cost. Hence, ARPA will converge to a solution and terminate in finite time slots.

ACKNOWLEDGEMENT

This work was supported in part by project "Corfu Virtual Exhibition Site for Tourism-Culture-Environment" (MIS 5031252), which is partially funded by European and National Greek Funds (ESPA) under the Regional Operational Programme "Ionian Islands 2014-2020".

REFERENCES

[1] Subashini Subashini and Veeraruna Kavitha. "A survey on security issues in service delivery models of cloud computing". In: *Journal of network and computer applications* 34.1 (2011), pp. 1–11.

[2] Wenwu Zhu, Chong Luo, Jianfeng Wang, and Shipeng Li. "Multimedia cloud computing". In: *IEEE Signal Processing Magazine* 28.3 (2011), pp. 59–69.

[3] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor CM Leung, and Cheng-Hsin Hsu. "A survey on cloud gaming: Future of computer games". In: *IEEE Access* 4 (2016), pp. 7605–7620.

[4] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. "Leveraging cloudlets for immersive collaborative applications". In: *IEEE Pervasive Computing* 12.4 (2013), pp. 30–38.

[5] Ryan Shea, Di Fu, and Jiangchuan Liu. "Cloud gaming: Understanding the support from advanced virtualization and hardware". In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.12 (2015), pp. 2026–2037.

[6] Zheng Xue, Di Wu, Jian He, Xiaojun Hei, and Yong Liu. "Playing high-end video games in the cloud: A measurement study". In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.12 (2014), pp. 2013–2025.

[7] Yichao Jin, Yonggang Wen, and Kyle Guan. "Toward cost-efficient content placement in media cloud: Modeling and analysis". In: *IEEE Transactions on Multimedia* 18.5 (2016), pp. 807–819.

[8] Qiumin Lu, Jianguo Yao, Zhengwei Qi, Bingsheng He, et al. "Fairness-efficiency allocation of cpu-gpu heterogeneous resources". In: *IEEE Transactions on Services Computing* (2016).

[9] Ivan Slivar, Mirko Suznjevic, and Lea Skorin-Kapov. "Game Categorization for Deriving QoE-Driven Video Encoding Configuration Strategies for Cloud Gaming". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.3s (2018), pp. 1–24.

[10] Hao Chen, Xu Zhang, Yiling Xu, Ju Ren, Jingtao Fan, Zhan Ma, and Wenjun Zhang. "T-Gaming: A Cost-Efficient Cloud Gaming System at Scale". In: *IEEE Transactions on Parallel and Distributed Systems* 30.12 (2019), pp. 2849–2865.

[11] Mohaddeseh Basiri and Abbas Rasoolzadegan. "Delay-aware resource provisioning for cost-efficient cloud gaming". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.4 (2016), pp. 972–983.

[12] Yongqiang Gao, Lin Wang, and Jiantao Zhou. "Cost-Efficient and Quality of Experience-Aware Provisioning of Virtual Machines for Multiplayer Cloud Gaming in Geographically Distributed Data Centers". In: *IEEE Access* 7 (2019), pp. 142574–142585.

[13] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. "A survey on mobile edge computing: The communication perspective". In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2322–2358.

[14] Yunhua Deng, Yusen Li, Ronald Seet, Xueyan Tang, and Wentong Cai. "The server allocation problem for session-based multiplayer cloud gaming". In: *IEEE Transactions on Multimedia* 20.5 (2017), pp. 1233–1245.

[15] Yuhua Lin and Haiying Shen. "CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service". In: *IEEE Transactions on Parallel and Distributed Systems* 28.2 (2017), pp. 431–445.

[16] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. "A hybrid edge-cloud architecture for reducing on-demand gaming latency". In: *Multimedia systems* 20.5 (2014), pp. 503–519.

[17] Ping Lu, Quanying Sun, Kaiyue Wu, and Zuqing Zhu. "Distributed online hybrid cloud management for profit-driven multimedia cloud computing". In: *IEEE Transactions on Multimedia* 17.8 (2015), pp. 1297–1308.

[18] Athanasios Tsipis, Konstantinos Oikonomou, Vasileios Komianos, and Ioannis Stavrakakis. "Performance evaluation in cloud-edge hybrid gaming systems". In: *Third International Balkan Conference on Communications and Networking 2019 (BalkanCom'19)*. Skopje, North Macedonia. 2019.

[19] Nimrod Megiddo and Kenneth J Supowit. "On the complexity of some common geometric location problems". In: *SIAM journal on computing* 13.1 (1984), pp. 182–196.

[20] Pitu B Mirchandani and Richard L Francis. *Discrete location theory*. 1990.

[21] Atakan Aral and Tolga Ovatman. "A decentralized replica placement algorithm for edge computing". In: *IEEE transactions on network and service management* 15.2 (2018), pp. 516–529.

[22] Mohammad A Salahuddin, Jagruti Sahoo, Roch Glitho, Halima Elbiaze, and Wessam Ajib. "A survey on content placement algorithms for cloud-based content delivery networks". In: *IEEE Access* 6 (2017), pp. 91–114.

[23] Giuseppe Portaluri, Davide Adami, Andrea Gabbrielli, Stefano Giordano, and Michele Pagano. "Power consumption-aware virtual machine placement in cloud data center". In: *IEEE Transactions on Green Communications and Networking* 1.4 (2017), pp. 541–550.

[24] Meng Wang, Xiaoqiao Meng, and Li Zhang. "Consolidating virtual machines with dynamic bandwidth demand in data centers". In: *2011 Proceedings IEEE INFOCOM*. IEEE. 2011, pp. 71–75.

[25] Shvan Omer, Sadoon Azizi, Mohammad Shojafar, and Rahim Tafazolli. "A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers". In: *Journal of Systems Architecture* 115 (2021), p. 101996.

[26] Ao Zhou, Shangguang Wang, Bo Cheng, Zibin Zheng, Fangchun Yang, Rong N Chang, Michael R Lyu, and Rajkumar Buyya. "Cloud service reliability enhancement via virtual machine placement optimization". In: *IEEE Transactions on Services Computing* 10.6 (2016), pp. 902–913.

[27] Abdulaziz Alashaikh, Eisa Alanazi, and Ala Al-Fuqaha. "A Survey on the Use of Preferences for Virtual Machine Placement in Cloud Data Centers". In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–39.

[28] Mohammad Goudarzi, Huaming Wu, Marimuthu S Palaniswami, and Rajkumar Buyya. "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments". In: *IEEE Transactions on Mobile Computing* (2020).

[29] Lili Qiu, Venkata N Padmanabhan, and Geoffrey M Voelker. "On the placement of web server replicas". In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*. Vol. 3. IEEE. 2001, pp. 1587–1596.

[30] Sin-Shuen Cheung and David P Williamson. "Greedy algorithms for the single-demand facility location problem". In: *Operations Research Letters* 45.5 (2017), pp. 452–455.

[31] Kun Cao, Liying Li, Yangguang Cui, Tongquan Wei, and Shiyan Hu. "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing". In: *IEEE Transactions on Industrial Informatics* 17.1 (2020), pp. 494–503.

[32] Thomas Moscibroda and Rogert Wattenhofer. "Facility location: distributed approximation". In: *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 2005, pp. 108–117.

[33] Jagruti Sahoo, Mohammad A Salahuddin, Roch Glitho, Halima Elbiaze, and Wessam Ajib. "A survey on replica server placement algorithms for content delivery networks". In: *IEEE Communications Surveys & Tutorials* 19.2 (2016), pp. 1002–1026.

[34] Shangguang Wang, Jinliang Xu, Ning Zhang, and Yujiong Liu. "A survey on service migration in mobile edge computing". In: *IEEE Access* 6 (2018), pp. 23511–23528.

[35] Mirsaeid Hosseini Shirvani, Amir Masoud Rahmani, and Amir Sahafi. "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges". In: *Journal of King Saud University-Computer and Information Sciences* 32.3 (2020), pp. 267–286.

[36] Changchun Long, Yang Cao, Tao Jiang, and Qian Zhang. "Edge computing framework for cooperative video processing in multimedia IoT systems". In: *IEEE Transactions on Multimedia* 20.5 (2017), pp. 1126–1139.

[37] Hua-Jun Hong, De-Yu Chen, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. "Placing virtual machines to optimize cloud gaming experience". In: *IEEE Transactions on Cloud Computing* 3.1 (2014), pp. 42–53.

[38] Alexandros Doumanoglou, David Griffin, Javier Serrano, Nikolaos Zioulis, Truong Khoa Phan, David Jiménez, Dimitrios Zarpalas, Federico Alvarez, Miguel Rio, and Petros Daras. "Quality of expe- rience for 3-D immersive media streaming". In: *IEEE Transactions on Broadcasting* 64.2 (2018), pp. 379–391.

[39] Tuo Cao, Zhuzhong Qian, Kun Wu, Mingxian Zhou, and Yibo Jin. "Service Placement and Bandwidth Allocation for MEC-enabled Mobile Cloud Gaming". In: *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE. 2021, pp. 179–188.

[40] Adyson M Maia, Yacine Ghamri-Doudane, Dario Vieira, and Miguel F de Castro. "Optimized place- ment of scalable iot services in edge computing". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2019, pp. 189–197.

[41] Jonghwa Choi and Sanghyun Ahn. "Optimal Service Provisioning for the Scalable Fog/Edge Computing Environment". In: *Sensors* 21.4 (2021), p. 1506.

[42] Yusen Li, Yunhua Deng, Xueyan Tang, Wentong Cai, Xiaoguang Liu, and Gang Wang. "Cost-efficient server provisioning for cloud gaming". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.3s (2018), pp. 1–22.

[43] Amira Rayane Benamer, Nejib Ben Hadj-Alouane, and Khaled Boussetta. "Online Games Servers Placement in Fog Computing: an Hybrid Bio-inspired Approach". In: *2020 IEEE 45th LCN Symposium on Emerging Topics in Networking (LCN Symposium)*. IEEE. 2020, pp. 141–149.

[44] Yuan Zhang, Lei Jiao, Jinyao Yan, and Xiaojun Lin. "Dynamic Service Placement for Virtual Reality Group Gaming on Mobile Edge Cloudlets". In: *IEEE Journal on Selected Areas in Communications* 37.8 (2019), pp. 1881–1897.

[45] Derian Alencar, Cristiano Both, Rodolfo Antunes, Helder Oliveira, Eduardo Cerqueira, and Denis Rosário. "Dynamic Microservice Allocation for Virtual Reality Distribution with QoE support". In: *IEEE Transactions on Network and Service Management* (2021).

[46] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. "Quality of Experience (QoE)-aware placement of applications in Fog computing environments". In: *Journal of Parallel and Distributed Computing* 132 (2019), pp. 190–203.

[47] Mohammad Goudarzi, Marimuthu Palaniswami, and Rajkumar Buyya. "A distributed application placement and migration management techniques for edge and fog computing environments". In: *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE. 2021, pp. 37–56.

[48] Paridhika Kayal and Jörg Liebeherr. "Autonomic service placement in fog computing". In: *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE. 2019, pp. 1–9.

[49] Yuanzhe Li, Ao Zhou, Xiao Ma, and Shangguang Wang. "Profit-aware Edge Server Placement". In: *IEEE Internet of Things Journal* (2021).

[50] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, Inwoong Kim, Xi Wang, Hakki C Cankaya, Qiong Zhang, Weisheng Xie, and Jason P Jue. "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework". In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 5080–5096.

[51] Tero Lähderanta, Teemu Leppänen, Leena Ruha, Lauri Lovén, Erkki Harjula, Mika Ylianttila, Jukka Riekki, and Mikko J Sillanpää. "Edge computing server placement with capacitated location allocation". In: *Journal of Parallel and Distributed Computing* 153 (2021), pp. 130–149.

[52] Lu Zhao, Wenan Tan, Bo Li, Qiang He, Li Huang, Yong Sun, Lida Xu, and Yun Yang. "Joint Shareability and Interference for Multiple Edge Application Deployment in Mobile Edge Computing Environment". In: *IEEE Internet of Things Journal* (2021).

[53] Palash Roy, Sujan Sarker, Md Abdur Razzaque, Mohammad Mehedi Hassan, Salman A AlQahtani, Gianluca Aloi, and Giancarlo Fortino. "AI-enabled mobile multimedia service instance placement scheme in mobile edge computing". In: *Computer Networks* 182 (2020), p. 107573.

[54] Feifei Chen, Jingwen Zhou, Xiaoyu Xia, Hai Jin, and Qiang He. "Optimal application deployment in mobile edge computing environment". In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE. 2020, pp. 184–192.

[55] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. "(How Much) Can Edge Computing Change Network Latency?" In: *2021 IFIP Networking Conference (IFIP Networking)*. IEEE. 2021, pp. 1–9.

[56] Feng Zeng, Yongzheng Ren, Xiaoheng Deng, and Wenjia Li. "Cost-effective edge server placement in wireless metropolitan area networks". In: *Sensors* 19.1 (2019), p. 32.

[57] C-A La, Pietro Michiardi, Claudio Casetti, C-F Chiasserini, and Marco Fiore. "A lightweight distributed solution to content replication in mobile networks". In: *2010 IEEE Wireless Communication and Networking Conference*. IEEE. 2010, pp. 1–6.

[58] Sharrukh Zaman and Daniel Grosu. "A distributed algorithm for the replica placement problem". In: *IEEE Transactions on Parallel and Distributed Systems* 22.9 (2011), pp. 1455–1468.

[59] Panagiotis Pantazopoulos, Merkouris Karaliopoulos, and Ioannis Stavrakakis. "Distributed placement of autonomic internet services". In: *IEEE transactions on parallel and distributed systems* 25.7 (2013), pp. 1702–1712.

[60] Eleni Kavvadia, Spyros Sagiadinos, Konstantinos Oikonomou, Giorgos Tsioutsiouliklis, and Sonia Aïssa. "Elastic virtual machine placement in cloud computing network environments". In: *Computer Networks* 93 (2015), pp. 435–447.

[61] Ryan Shea, Jiangchuan Liu, Edith C-H Ngai, and Yong Cui. "Cloud gaming: architecture and performance". In: *IEEE network* 27.4 (2013), pp. 16–21.

[62] Athanasios Tsipis, Konstantinos Oikonomou, Vasileios Komianos, and Ioannis Stavrakakis. "QoE-Aware Rendering Service Allocation in Fog-Assisted Cloud Gaming Environments". In: *2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. IEEE. 2020, pp. 1–8.

[63] Guy Almes, Sunil Kalidindi, Matthew Zekauskas, and Al Morton. "A one-way delay metric for IP performance metrics (IPPM)". In: *IETF, January* 10 (2016).

[64] Klervie Toczé, Johan Lindqvist, and Simin Nadjm-Tehrani. "Performance study of mixed reality for edge computing". In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. 2019, pp. 285–294.

[65] András Varga and Rudolf Hornig. "An overview of the OMNeT++ simulation environment". In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. 2008, pp. 1–10.

[66] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang. "Optimal edge user allocation in edge computing with variable sized vector bin packing". In: *International Conference on Service-Oriented Computing*. Springer. 2018, pp. 230–245.

[67] Hao Yin, Xu Zhang, Hongqiang H Liu, Yan Luo, Chen Tian, Shuoyao Zhao, and Feng Li. "Edge provisioning with flexible server placement". In: *IEEE Transactions on Parallel and Distributed Systems* 28.4 (2016), pp. 1031–1045.

AUTHORS

Athanasios Tsipis received in 2015 his B.Sc. degree in Informatics from the Department of Informatics, Ionian University, Greece. In 2017 he obtained his M.Sc. degree in Computer Science and Information Systems, for which he was awarded a tuition-free scholarship, while in 2021 he received his Ph.D. degree in Informatics, both from the same Department. His research interests concentrate on multimedia cloud computing, cloud gaming, fog/edge computing, network optimization, performance issues in IoT and wireless networks, facility location allocation, service placement, etc. Since 2015 he has been an active member and researcher of the "Networks, Multimedia and Security Systems Laboratory" (NMSLab), participating in various EC research and local development projects. Currently, he serves as a teaching assistant at the Department of Digital Media and Communication, as well as an academic scholar at the Department of Informatics, at Ionian University. He has been a reviewer of numerous conferences and journals, and in 2021 he was the recipient of the best paper award from the 26th IEEE Symposium on Computers and Communications.

Konstantinos Oikonomou received his M.Eng. degree in Computer Engineering and Informatics from the University of Patras in 1998. In September 1999 he received his M.Sc. degree in Communication and Signal Processing from the Electrical Department, Imperial College (London). He received his Ph.D. degree in 2004 from the Informatics and Telecommunications Department, University of Athens, Greece.

His Ph.D. thesis focuses on medium access control policies in ad hoc networks. He has served as the Dean of the Faculty of Information Science and Informatics of the Ionian University, Corfu, Greece. Since 2006 he is a faculty member in Computer Networks, currently a Professor, at the Department of Informatics, at the same University. He has also served as the Head of the Department and Director of the Postgraduate Studies. Between December 1999 and January 2005 he was employed at Intracom S.A, as a research and development engineer. His research interests involve medium access control in ad hoc networks, performance issues in wireless networks, information dissemination, service discovery, facility location, energy consumption in wireless sensor networks, and network cost reduction in cloud computing environments. Professor K. Oikonomou has a long experience in wireless systems and has been coordinating a number of EC research projects in the computer networks area, and various local development projects (e.g., virtual worlds). He is currently a member of the editorial boards of Computer Networks Journal (Elsevier) and Journal on Future and Evolving Technologies (ITU). He has been a reviewer and TPC member of numerous conferences and journals, and he holds an award for the best paper from the Hawaii International Conference on System Service, as well as an award for the best paper from the 26th IEEE Symposium on Computers and Communications.