Journal of Grid Computing manuscript No.
(will be inserted by the editor)

Addressing Application Latency Requirements through Edge Scheduling

Atakan Aral · Ivona Brandic · Rafael Brundo Uriarte · Rocco De Nicola · Vincenzo Scoca

Received: 14 December 2018 / Accepted: 12 July 2019

Abstract Latency-sensitive and data-intensive applications, such as IoT or mobile services, are leveraged by Edge computing, which extends the cloud ecosystem with distributed computational resources in proximity to data providers and consumers. This brings significant benefits in terms of lower latency and higher bandwidth. However, by definition, edge computing has limited resources with respect to cloud counterparts; thus, there exists a trade-off between proximity to users and resource utilization. Moreover, service availability is a significant concern at the edge of the network, where extensive support systems as in cloud data centers are not usually present. To overcome these limitations, we propose a score-based edge service scheduling algorithm that evaluates network, compute, and reliability capabilities of edge nodes. The algorithm outputs the maximum scoring mapping between resources and services with regard to four critical aspects of service quality. Our simulation-based experiments on live video streaming services demonstrate significant improvements in both network delay and service time. Moreover, we compare edge computing with cloud computing and content delivery networks within the context of latency-sensitive and data-intensive applications. The results suggest that our edge-based scheduling algorithm is a viable solution for high service quality and responsiveness in deploying such applications.

Keywords Edge Computing · Scheduling · Live Streaming

1 Introduction

Cloud Computing is currently the predominant hosting solution for internet services due to the economical and infrastructural advantages it provides. Indeed, the massive pool of redundant resources characterizing cloud data centers benefits from significantly lower marginal

All authors have contributed equally and are listed alphabetically.

A. Aral · I. Brandic · R.B. Uriarte
Vienna University of Technology, Austria
E-mail: {name.surname}@tuwien.ac.at

R. De Nicola · V. Scoca
IMT School for Advanced Studies Lucca, Italy
E-mail: {name.surname}@imtlucca.it

costs due to economies of scale and guarantees high level of reliability and dynamism, which allow the providers to scale up/down the allocated resources based on current needs. As a significant challenge to this centralized paradigm, rapid progress in smart devices and network technologies has enabled new categories of internet-based services with strict end-to-end latency requirements and with a vast amount of data to process. Such near real-time and data-intensive services include live and 360 degrees video streaming, online gaming, intelligent traffic control, and smart power grids. Since centralized deployment solutions fail to satisfy strict latency requirements and network architectures will soon become incapable of handling a massive amount of data communication, more geographically distributed approaches are necessary (Satyanarayanan, 2017). Indeed, many cloud providers already enrich their offers with distributed deployments solutions including Content Delivery Networks (CDN) and Edge Computing.

Edge Computing proposes to place computation and storage capabilities at the edge of the network in the form of micro scale data centers as an extension of massive scale cloud data centers (Shi et al., 2016). This is a highly promising solution for the aforementioned latency-sensitive services since it allows the deployment of services in close proximity of their end users to meet response time requirements. Early work on the impact of Edge Computing on service deployment (Hu et al., 2016; Bittencourt et al., 2017) provides insights its effectiveness in terms of end-to-end service latency, which ensures a higher quality of service for end users. However, such benefits can be achieved only if service instances are effectively scheduled on the nodes that satisfy their requirements on latency, bandwidth, computation capacity, and reliability. In this regard, contextual information (i.e., user, application, computation, and network) must be taken into account to find an optimal service placement (Wang et al., 2017). This would not only maximize the quality of user experience by decreasing end-to-end latency, but also minimize the backbone network traffic since most data communication would be local.

Currently, only a few works, mainly in the area of Internet-of-Things, consider service scheduling for Edge Computing. Existing scheduling solutions borrowed from similar paradigms, in particular cloud and CDN, are subject to infrastructure limitations in the context of Edge Computing, which renders them unsuitable for meeting the requirements of latency-sensitive services. More specifically, cloud-based solutions depend on the massive pool of resources, homogeneous network conditions for compute nodes, as well as high reliability and scalability. CDN paradigm is similar to Edge Computing in the sense that the resources are distributed and close to users. However, CDN are designed for data-intensive services rather than for computing intensive ones and computations are still offloaded to the cloud (Bilal and Erbad, 2017). Hence, CDN scheduling solutions ignore processing capabilities.

In this paper, we propose a score-based edge scheduling framework specifically designed for latency-sensitive, computational and data-intensive services at the edge of a network. Proposed scheduling mechanisms are applicable to virtual machines (VM) as well as to more light-weight implementations such as containers; for the sake of brevity in the rest of the paper, we use the term VM for both. For each service instance, our approach identifies the VM type with the computational and network capabilities that minimizes the response time for end users without reserving resources in excess. The algorithm first evaluates the eligibility of available VM types on the edge nodes for hosting a given service by considering network latency, bandwidth, processing power, and reliability. Then, it schedules services on the most appropriate VMs according to the overall eligibility scores, to guarantee optimal service quality. We validate our approach by considering a live video streaming scenario and evaluating how the different deployment solutions, namely Cloud, CDN, and Edge, affect user response time. Our results clearly demonstrate that an Edge-based deploy-

ment can effectively improve user response time. Our main contributions are: (i) a novel Edge scheduling framework for latency-sensitive services; and (ii) an evaluation of different deployment solutions and scheduling approaches for latency-sensitive services.

This paper is an extended and revised version of (Scoca et al., 2018). Here, the scheduling algorithm takes availability into account and the proposed solution considers also the integration of Edge with Cloud, which turns out to be useful especially when Edge data centers are saturated. All experiments are revised to take into account these new perspectives. In the following section, we motivate our work by discussing the benefits of edge-based deployment for latency-sensitive live video streaming services. In Sec.3, we present our scheduling approach and in Sec. 4 we introduce the experimental setup for the evaluation of this approach. We present and discuss numerical results in Sec. 5, whereas we survey the related literature in Sec. 6. We conclude the paper and discuss future directions in Sec. 7.

2 Motivation

2.1 Use Case Scenario

Smart-phones are widely used for recording and sharing live events thanks to their ubiquitousness and improved video recording capabilities. In mobile live video streaming scenario, a video recorded by a mobile device is live streamed globally to any number of mobile or desktop audience. The success of pioneer services such as Meerkat that have been used by journalists, lecturers, marketing firms, event organizes, etc. has attracted the interest of large companies such Twitter (Periscope), Facebook (Facebook Live), Google (Youtube Mobile Live), and IBM (Ustream, IBM Cloud Video). Fig. 1 depicts the major components and workflow of a live video streaming service, which is discussed in detail in the rest of this section.

As the foremost operation, raw input media from the camera is encoded into a high-quality stream by either the local camera encoder or an external one, running on a server preferably close to the streaming venue. Transcoding is the next step, where new stream instances in different resolutions and bit-rates are created on the fly. As a result, the audience with different device configurations in terms of network bandwidth, screen resolution and size can stream a suitable stream and experience a high quality of service. Each transcoded stream then undergoes the packaging operation. In this step, the streams are split into chunks of a size that is manageable from the network communication perspective. Packaging operation is carried out based on a streaming protocol (e.g., HLS or HDS), which defines the chunk length, codecs, container, etc. Finally, the video chunks are delivered to the corresponding audience devices, which are responsible for sorting and merging them, and playing back video content through a media player.

Deployment of the above described operations on the widely distributed architecture of Cloud, CDN, and Edge nodes plays a critical role in the engagement of the audience with the video stream. More specifically, it has a significant impact on the various user engagement metrics such as join time (i.e., the time between the connection establishment and the beginning of the playback), buffering ratio (i.e., the percentage of buffering time in a video) and video quality (Dobrian et al., 2011). Less computation capability at the deployed servers and longer geographical distance translates to either higher join time and buffering ratio or lower video quality, which decreases user engagement.

The most widely implemented solution at the moment by the service providers is the cloud-based deployment. This allows fast scalability of resources in the face of volatile

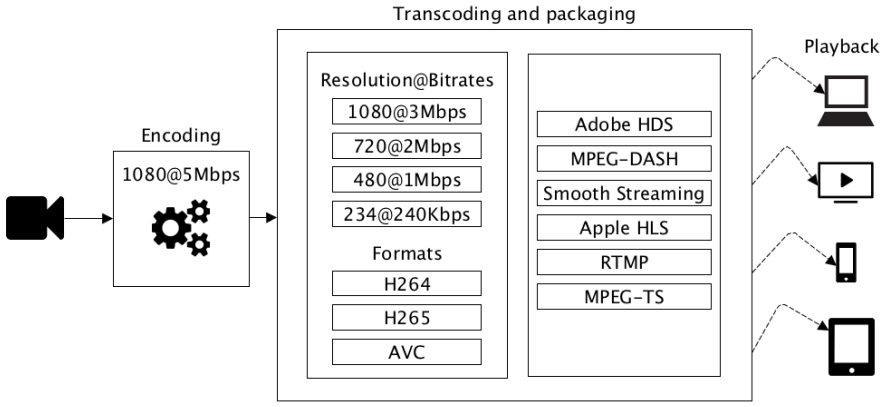


Fig. 1: Live video streaming workflow.

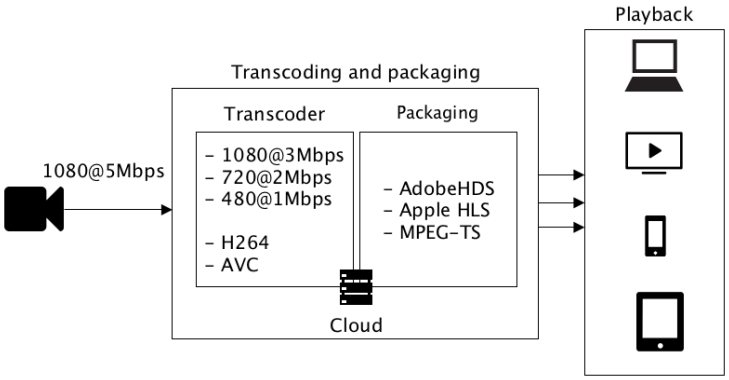


Fig. 2: Cloud-based solution for live streaming services.

workloads caused by, for example, flash crowds. Fig. 2 depicts the simplest form of cloud-based deployment, where transcoding and packaging operations are carried out in a cloud data center and directly delivered to the audience from there. As a result, transcoding and packaging operations enjoy seemingly unlimited resource capacity and elasticity of cloud data center. Even though, video can be processed quite efficiently in this scenario, delivery of the media content between a centralized cloud data center and possibly globally distributed users is the bottleneck. This is because current wide area networks can provide only best effort guarantee for packet delivery. Considering the geographical distance and high number of networks hops along the route, link congestion is highly probable, which leads to packet loss, jitter and low throughput and consequently a low quality of video experience.

To overcome above described issues, the state-of-the-art cloud deployment strategies involve the use of content delivery networks (CDNs) as demonstrated in Fig. 3. Here, another level of caching is added between the audience and cloud data center. Whereas media processing still occurs in the cloud, distribution is offloaded to a CDN provider, which maintains a network of cache servers in close proximity to the end users. Each user request is then redi-

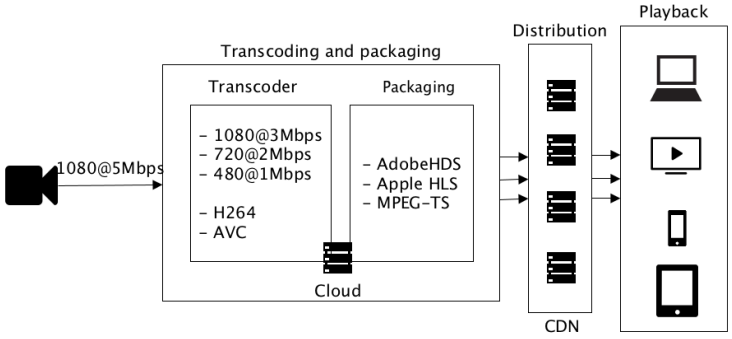


Fig. 3: Delivery networks for live streaming services.

rected to the closest cache server with the required version of the video stream. As a result, bandwidth consumption and network delay between the video processing and the audience are reduced. Cloud-based and CDN supported video streaming are successfully employed by many video providers that have fairly static media content in centralized repositories. However, this approach has significant shortcomings when it comes to live videos originating from end users. First, problems with the network communication between the recording device and the cloud data center are still present. Especially in cases that the recorder and audience are in close proximity (e.g., within the same city), transferring the stream to a remote data center causes unnecessary network traffic. Second, continuous update of the CDN servers due to the transient nature of live video streaming is inefficient in terms of network bandwidth and monetary cost. We argue that CDN architecture is insufficient for highly demanding live video streaming usage.

In this work, we propose an edge-based live video streaming architecture as given in Fig. 4 in order to eliminate above described disadvantages of cloud and CDN-based deployments. Here, raw video is encoded either locally or at the edge node that is closest to the recorder. Afterwards, it is distributed to a set of edge nodes in close proximity to the audience for the execution of transcoding and packaging operations. Different from the cloud-based video processing, where streams in all possible combinations of bit-rates, resolutions, and protocols have to be generated; only the requirements of the local audience, who are serviced by a particular edge node, need to be satisfied. This results in efficient utilization of limited computing capacity of edge servers. Moreover, they have a supplementary task of caching the video chunks similar to the CDN servers. Hence, each audience is served by the closest edge node, which generates the stream in the required format.

In summary, encoding, transcoding, and packaging of the live video stream can be carried out close to the recorder and the audience. Consequently, additional network delays and detours from the shortest route are mostly eliminated. Moreover, similar media processing performance to cloud computing can be achieved by means of parallelization. Load on the wide area network is also alleviated since the majority of the data transfer occurs in the local area, where edge nodes are usually accessible by the user devices through high-bandwidth wireless connections. All these improvements result in higher quality of experience in playback and increase user engagement.

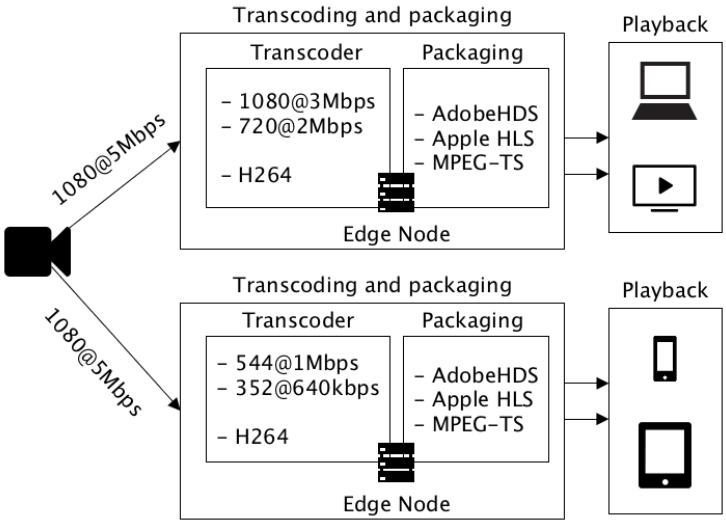


Fig. 4: Edge-based platform for live streaming services.

2.2 Edge Business Models

Despite the potential of Edge computing, the business models of this emerging paradigm are complex because of the different types of providers, the mobile nature of the services and their limited scope (e.g., ephemeral or spike processing demands). Many different players can take part in this market: cloud providers, which want to expand their services and infrastructure; Internet service providers (ISPs), which can install computational resources in their existing infrastructure; and so-called, *prosumers*, i.e., users with spare computational resources, who can, at the same time, consume and provide resources. All these actors face significant challenges to actually enter this market. Cloud providers profiting from the economy of scale and the centralization of services may have to change their business model to interact with geographically distributed small data centers. ISPs that do not have provisioning of computational resources as core business need to consider them. *Prosumers* of edge computing need to acquire the skills and the tools to properly manage services and avoid resorting to larger providers because of the guarantees they offer.

In addition, service models for edge computing are manifold. The infrastructure and platform as a service models may work well in scenarios where the same service is provided to many users, like, e.g., the live video streaming scenario covered in this paper. In these scenarios, VMs can host services for groups of users. However, their use is not ideal for applications related to a single user since VMs' initialization and migration time is relatively high. For example, when edge computing is employed to complement the computational power of user's device to deal with the CPU intensive part of an application, users can share the same resources (bare-metal servers, VMs, etc). Alternatively, providers need to deploy lighter solutions, e.g., containers.

Since in edge environments users can be mobile or require seamless service handover, the single provider model used in cloud is only applicable if large providers own geographically distributed resources covering the needs of most of its consumers. A solution to this

problem could be that brokers agree with local resource providers (e.g., private cloud owners), to acquire a share of their infrastructure and create a resource network to be sold to edge consumers. An alternative could be the creation of edge open markets with low market entry barriers to increase the number of providers and leverage the adoption of edge computing, while providing a single interface for edge consumers. To this end, the capacity of blockchain and smart contracts to create a trust layer between participants could be very useful. Indeed, blockchain and smart contracts have attracted attention from industry and academia. Several projects are emerging on the area, e.g., (Tuli et al., 2019; Stanciu, 2017), but there are still many open gaps, such as, defining QoS and pricing policies (Uriarte et al., 2019, 2016a; Scoca et al., 2017), comparing resource offers and providing seamless handover. For an overview of the challenges and advantages of these solutions, we refer to (Uriarte and De Nicola, 2018). The scheduling algorithm we are proposing can be used in any of these models and by any types of providers. It might only be necessary to introduce some small extensions to take into account specific aspects, e.g., brokers might need to consider also costs.

3 A Scheduling Framework for Edge Computing

This work uses the terminology of (Hu et al., 2015), which defines edge networks as geographically distributed nodes, relatively close to Radio Access Technology (RAT) Base Stations (BSs). These stations provide to users and edge nodes access to the core network. Each node, in our context, is a micro data center that grants access to its resources through different types of Virtual Machines (VMs).

Considering that we focus on latency-sensitive services, the most important service requirement is response time, i.e., the time between a user request and its reply. In this scenario, the response time is mainly composed of processing time and network delay. The former refers to the time elapsed between the user request and its arrival in the edge node providing the service. The queuing, user-node distance, and overhead of each hop in the network route and the available bandwidth are the main metrics that compose the network delay. Processing time, instead, is calculated based on the VM characteristics and service specifications and refers to the time necessary to compute a user request, which depends on the type of VM it is executed and on the service specifications.

We design a resource reservation and scheduling framework, which considers the edge computing characteristics in the scheduling process to improve the experience of latency-sensitive application users. Alg. 1 and Fig. 5 illustrate the main steps of our methodology. For each service, it groups the users based on their location and requirements; evaluates the network quality of each node, in particular, its connectivity and bandwidth; evaluates the VM's resources and availability with respect to the service requirements; and finally, based on the combination of these evaluations, schedules the service. These steps are detailed in the rest of this section with references to the line numbers in Alg. 1.

3.1 User Clustering

Considering the large audience of live video streams, it would be extremely time and resource consuming to compute scores for each individual user. Fortunately, consumers and producers of many edge services are known to exhibit geographical locality (Aral and Ovatman, 2018), thus their compute and network requirements are similar. In this work, users

Algorithm 1: Latency-Sensitive Scheduling

Data: Service s to be scheduled
Data: Service provider's latency constraint t
Data: Location of the users $U = \{u_1; u_2; \dots; u_m\}$
Data: Nodes $N = \{n_1; n_2; \dots; n_k\}$
Data: Location of the nodes $L = \{l_1; l_2; \dots; l_k\}$
Data: Types of VMs $VM = \{v_{i;1}; v_{i;2}; \dots; v_{i;q}; \forall i \in N\}$
Data: Array Q of quality scores for each VM $v \in VM$
Result: The vm_s scheduled for s.

1 **begin**
2 Define a set of users' groups $G = \{g_i; |u_j - l_i| < e \quad \forall u_j \in U; \forall l_i \in L\}$;
3 **forall** $n \in N$ **do**
4 **forall** $g \in G$ **do**
5 Estimate the network path $p_{g;n}$ with the lowest latency $\tilde{t}_{g;n}$ between g and n;
6 Estimate the available bandwidth $\tilde{bw}_{g;n}$ between users in g and the node n and the required bandwidth bw_g on $p_{g;n}$;
7 Calculate the connectivity score $q_{n;l;g} = u$;
8 Calculate the bandwidth score $q_{n;bw;g} = \text{BandwidthScore}(bw_g; \tilde{bw}_{g;n})$;
9 $q_{n;l} = \frac{\sum_{i=1}^{n} |g_i| q_{n;l;g_i}}{\sum_{i=1}^{n} |g_i|}$;
10 $q_{n;bw} = \frac{\sum_{i=1}^{n} |g_i| q_{n;bw;g_i}}{\sum_{i=1}^{n} |g_i|}$;
11 **forall** $v \in n$ **do**
12 $q_{v;res} = \text{ComputingScore}(w; \bar{w}_v)$;
13 $q_{v;av} = \text{AvailabilityScore}(av; \bar{av}_v)$;
14 Calculate the q_v quality score $Q[q_v] = \frac{4}{\frac{1}{q_{v;res}} + + \frac{1}{q_{v;av}} + \frac{1}{q_{n;l}} + \frac{1}{q_{n;bw}}}$
15 $vm_s = Scheduler(Q)$;
16 **return** vm_s;

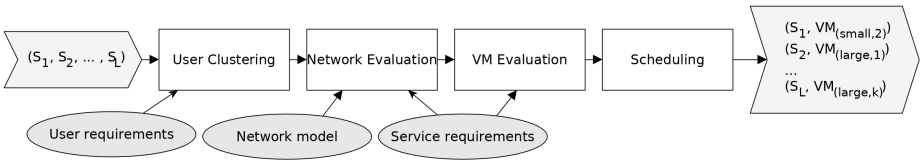


Fig. 5: The scheduling framework.

with similar requirements in close proximity are clustered into user groups in order to reduce the complexity and consequently the time of VM evaluation. In this way, computing only a single overall score for each group of users is sufficient. We first assign each user to the geographically closest edge node, and then we cluster the users who are assigned to the same node and have similar compute requirements into a group (line 2). We use three levels of CPU consumption (low, medium, and high), which represent, for example, different encoding length and video resolution requirements of smartphones, tablets, and laptops in the live streaming use case.

3.2 Network Evaluation: Connectivity and Bandwidth

The network evaluation is divided into three main phases: calculating the latency between the previous defined groups and the edge nodes; defining a connectivity score; and a bandwidth score.

In our solution, we assume that edge nodes are connected to each other through a network and that users connect to the closest BS. In many cases, however, the service may not be scheduled in the closest edge node to its user. For example, the edge node might node not have the type of resources required by the service or it might be overloaded. We, therefore, need to take into account the latency between groups and the edge nodes. To this aim, we: (i) represent the edge network as a weighted graph, where each node is a data center in the network, the link between nodes define the connection between data centers and the link weight describes the estimated network latency between them; and (ii) devised an user-based approach that, for every group of users g, detects the lowest latency path from a node n to g by relying on the Dijkstra's algorithm (line 5). This method is valid since network latency can be estimated using many different techniques, which range from active monitoring by sending probes between servers and then measuring the latency, to passive monitoring that capture information from the network device about network paths (Yu et al., 2015).

With the *connectivity score* $q_{n;l} \in [0;1]$ we define the quality of the connectivity between each user and VM v by evaluating the network routes between user groups and the node n of v. The input for this process is the previously computed latency. The evaluation of the delay, executed for every path connecting a user group g to the n node, is computed by a utility function previously defined by the provider, for example, a sigmoidal function that assigns values close to 1 if the network delay of the path is under a given value and 0 if over. The path-based evaluation produces a set of quality scores $\{q_{n;l;g_1}; q_{n;l;g_2}; \dots; q_{n;l;g_N}\}$, later used to calculate the connectivity final score $q_{n;l}$ as the mean value of the scores of groups using this path (line 9).

One of the main factors affecting the overall service quality, as mentioned in Sec.2, is the available bandwidth of the paths between users and the VM. The *bandwidth score* $q_{n;bw} \in [0;1]$ represents the available bandwidth for each n. We calculate a *bandwidth score* $q_{n;bw;g_i} \in [0;1]$ for each shortest path from n to each user group g_i. Similar to the delay evaluation process, we calculate the final bandwidth quality score $q_{n;bw}$ by averaging the single path quality scores $q_{n;bw;g_i}$, weighted by the number of users in each group (line 10).

3.3 Virtual machine evaluation: Resources and Availability

To measure the *VM resource evaluation* we take into account the expected overall service load and the load a VM type can handle. For latency-sensitive contexts, estimating fewer resources than actually required by the application execution may increase considerably the overall response and the processing time. Our approach computes a score $q_{v;res} \in [0;1]$ to evaluate the computational resources of a VM v, similarly to the network evaluation process, by using a utility function defined by the provider, which compares the number of user requests that v can be coped with the overall number of requests expected (line 12). Other utility functions can also be used, for example, the similarity of the new service with other services running in the same host (Uriarte et al., 2016b) or the performance of the type of the new service in the target hardware.

Another critical factor for service performance at the edge of the network is the reliability of computational resources. Edge servers are known to be more prone to failures in

comparison to cloud counterparts (Aral and Brandic, 2017). In this work, we consider transient failures at edge nodes such as power outage, system restart, memory overflow, etc. We model the reliability of each node as its historical *availability rate*, which is shown to be sufficiently accurate assuming the failures are independent (Aral and Brandic, 2018). Desired availability is taken as 100% in order to guarantee the computation capacity promised by VM resource evaluation. A user-defined utility function computes an availability score $q_{v;av} \in [0;1]$ (line 13).

3.4 Scheduling Latency-Sensitive Services

Given the set $s \in S$ of services, the provider P schedules each s on VM type $v \in V$, which can guarantee the end user service quality. First, we compute for each VM type an overall quality score $q_{s;v}$ by calculating the harmonic mean of the connectivity, bandwidth, resource and availability scores (line 14). Although the harmonic mean is similar to the arithmetic mean since they both give the same weight to two scores when they are similar, it increases the importance of smaller values when the gap between two values increases. This guarantees that very high and very low scores VMs are penalized, which favors a better balance between the network and computational resources. The output of the VM evaluation is the set $Q = \{q_{v_1}, \ldots, q_{v_k}\}$ of quality scores (line 15).

Then, the scheduling process maximizes the overall quality of the selected VMs, guaranteeing the service quality to end users. This process was defined as a binary integer programming optimization problem as shown in the formulation below. $x_{s;v}$ are binary variables that take true value only if service s is scheduled to VM type v. The coefficients $q_{s;v}$, on the other hand, are the VM quality scores calculated by the VM evaluation algorithm, which represent the suitability of VM type v for service s. The quality of the final scheduling is maximized by the cost function (I).

$$\underset{x}{\text{maximize}} \quad \sum_{s \in S} \sum_{v \in V} q_{s;v} x_{s;v} \qquad \text{(I)}$$

$$\text{s. to} \quad \sum_{v \in V} x_{s;v} = 1 \qquad \forall s \qquad \text{(II)}$$

$$\sum_{s \in S} x_{s;v} <= k_v \qquad \forall v \in V_n \qquad \text{(III)}$$

where

$$x_{s;v} = \begin{cases} 1 \text{ if } s \text{ is scheduled on } v \\ 0 \text{ otherwise} \end{cases}$$

Additionally, constraint (II) guarantees that each service is scheduled to a single VM, whereas (III) guarantees that the number of provisioned VMs of type v cannot exceed the number available instances of that type, k_v.

4 Simulation Environment

We developed a simulation environment by extending the EdgeCloudSim framework (Sonmez et al., 2017), which itself is an edge computing extension to the widely used CloudSim toolkit (Calheiros et al., 2011). We experiment with several deployment scenarios and scheduling policies to cover different aspects of the problem. Moreover, we employ real-world

resource and workload characteristics, where possible, in order to obtain realistic results. When real-world traces are unavailable, we resort to synthetic generation via methods that are shown to be effective by previous studies. The rest of this section describes the simulation environment in detail.

4.1 Deployment Scenarios

We simulate the live video streaming scenario described in Sec. 2 as a use case for the proposed scheduling framework. In this scenario, computation capability and network latency characteristics of the chosen nodes for the transcoding, packaging, and delivery of live video have a strong impact on the QoS perceived by the users. We also compare the performance of Cloud, CDN, Edge and we consider a hybrid solution, where both Cloud and Edge resources are available.

Cloud: This is the centralized deployment, where the original video is sent to a Cloud data center and all video processing (i.e., encoding, transcoding, packaging) is carried out in powerful servers. Video segments are also distributed from the data center to the users, who request the live stream. Considering the huge computation capacity of a Cloud data center, resources are modeled as an infinite set of VM instances with specifications taken from the Amazon Web Services (AWS) m2.4xlarge as reported in Tab. 1. The network communication between the user and the Cloud data center is modeled with a single 1 Gbps link. This represents the aggregation of multiple links from user access BSs to the Cloud. To estimate the communication delay of this link, we made ICMP requests from hosts in Lucca, Italy and the AWS instances in the same time zone and averaged the experienced latency. In our experiments, we use the communication delay $d_{u;c} = 0.09s$ to model the latency due to the queuing and processing operations as well as the physical distance between the user u and the Cloud data center c.

Content delivery network (CDN): Also in this scenario, the live video is transmitted to and processed in the Cloud data center. However, distributed cache servers store the video segments for future requests. Content distribution follows the policy proposed by Pathan and Buyya (2007). Here, a user request is first redirected to the closest cache server which returns the content if it is already cached. As a result, an additional delay to the Cloud is avoided. If the requested content is not available in the cache, the request is forwarded to the Cloud. In this case, the response is both sent to the user and stored at that cache server. We consider a three-tier network architecture of the users, geographically distributed CDN nodes, and an origin server in the Cloud data center. In this scenario, the origin and replica (CDN)

Table 1: Virtual machine specifications.

	m1.large	m1.xlarge	m2.4xlarge	i3.large
Number of CPUs	4	8	26	2
CPU (MIPS)	2400	4800	20000	2400
RAM (GB)	8	16	70	15
Storage (GB)	2x420	4x420	2x840	unlimited
Price (USD/h)	0.17	0.35	0.98	0.15

servers have different purposes and correspondingly different hardware configurations. Origin server is of type m2.4xlarge similar to the Cloud scenario, whereas replica servers, being intended for delivering content, have storage optimized i3.large specifications as shown in Tab. 1. Regarding the network characteristics, CDN nodes feature substantial inter-node and node-to-cloud distance because they are geographically distributed and co-located with network nodes, such as ISP point of presences (PoPs) or at internet exchange points (IXPs). However, they are in close proximity to the users, which reduces the user-replica network latency. Hence, we define a communication delay of $d_{u;r} = 0{:}013s$ between the user and the CDN node along with a $d_l = 0{:}03s$ for each hop on the path between the CDN and the origin server. $d_{u;r}$ is approximated by sending a set of ICMP requests from a host to a server distant around 300km over a 4 hops connection. Access bandwidths of CDN servers are generated as a Pareto distribution with a mean value of $\mu = 500Mbps$, whereas the links between the Cloud and CDN nodes are of higher capacity with $\mu = 750Mbps$, since we assume that they are directly connected to the ISP backbone. Underlying network topology is described in Sec. 4.3.

Edge: In this scenario, entire service is deployed on the edge computing servers as described in Sec.2. Hence, the cloud data center is only responsible for the management of the edge nodes, which execute both video processing and distribution operations. We deploy 20 Edge nodes that are co-located with BSs. At each node, we allow 10 VMs of type either m1.large or m1.xlarge to be instantiated in order to reflect the limited computing power of Edge nodes. The access bandwidths of edge nodes are modeled as a Pareto distribution with average value $\mu = 375Mbps$. Considering the locality of Edge nodes, we assume that Edge nodes are in close proximity not only to the user but also to other nodes. Hence, we define high-speed connections through one-hop dedicated links between Edge nodes, where bandwidth values are from a Pareto distribution with a mean value of $\mu = 400Mbps$, and latency from a uniform distribution $U[0{:}006; 0{:}009]$. In this case, we estimated the $d_{e_i;e_j}$ values as the average latency measured from a set of ICMP request sent over a single hop connection between two hosts within 40 km distance.

Hybrid: This scenario extends the Edge one by merging it with the Cloud approach. It considers, besides the 20 nodes available in the edge network, also a cloud data center in the actual service deployment. Intuitively, if there are not enough resources available in the reserved VM, our solution checks the availability of other VMs in the same host. In case it is not available, instead of using other edge data centers, which could affect the reservation scheme provided by the scheduler, it sends the service to a cloud data center. Overall, this approach provides a hybrid solution that integrates the virtually infinity computation power of the clouds to edge, which is used only in cases where the edge micro data centers are saturated. Computation and communication capabilities of the Edge and Cloud nodes are the same as their respectable scenarios described above.

4.2 Scheduling Scenarios

In addition to the Edge-based scheduling approach proposed in Sec. 3.4, we also implement a state-of-the-art Cloud-based scheduling approach (Duong et al., 2012) as a baseline.

Edge-based: We use the utility functions in Eq. 1-4 for the four scores described in Sec.3.3, namely the network delay, available bandwidth, VM resources, and VM availability. Utility score for the network delay between a user group g and a virtual machine v (Eq. 1) is computed via sigmoidal function S. Domain of the function is the ratio of the delay $d_{g;v}$ to the maximum tolerable delay $\tilde{d}$ which is set to $50ms$. Additive inverse indicates that smaller $d_{g;v}$ values are preferred. As a result, delays $d_{g;v}$ that are over the requirements $\tilde{d}$, receive scores close to 0, whereas the scores within the limits receive scores close to 1. Use of the sigmoidal function ensures that delays shorter that the requirements are not incentivized. Other functions are defined in a similar fashion. For the bandwidth score (Eq. 2), available bandwidth $B_{g;v}$ is compared to the bandwidth demand $\tilde{B}$, whereas for the VM resources (Eq. 3), the computing capability of v expressed in terms of requests per second RPS_v is compared to the expected number of requests $\tilde{W}$ based on the expected workload. Utility function for the availability (Eq. 4), on the other hand, compares the availability rate of the node, A_v, to the optimal case of 100% availability. In Fig. 6, the functions are plotted for varying resources availabilities given a request of maximum 50 ms latency, minimum 300 Mbps bandwidth, and capability to process at least 25 requests per second with 95% availability.

$$u_{v;d}(d_{g;v};\tilde{d}) = S\left(1 - \frac{d_{g;v}}{\tilde{d}}\right) \tag{1}$$

$$u_{v;B}(B_{g;v};\tilde{B}) = S\left(\frac{B_{g;v}}{\tilde{B}} - 1\right) \tag{2}$$

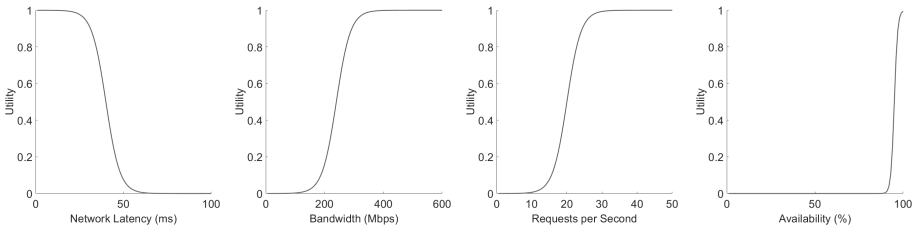
$$u_{v;RPS}(RPS_v;\tilde{W}) = S\left(\frac{RPS_v}{\tilde{W}} - 1\right) \tag{3}$$

$$u_{v;A}(A_v) = S\left(\frac{A_v}{100} - 1\right) \tag{4}$$

Cloud-based: There exist a plethora of task scheduling algorithms in the context of cloud computing. We implement the algorithm by Duong et al. (2012) called *FIXED provisioning algorithm* as a representative baseline for cloud-based scheduling. This algorithm estimates the minimum number of streaming engine instances required to meet a tolerable user waiting time. It proposes a M/M/V[5] queue to accurately model the system and proactively adjust the number of instances according to the workload prediction. We provide the pseudo-code of this algorithm in Alg. 2. Given the estimations for inter-arrival times and durations of the streaming requests, it begins with calculating an initial set of VMs with the minimum size (line 2). Then, t is defined as the minimum waiting time to be guaranteed based on the QoS requirements of the ASP (line 3). The algorithm increases the number of VM instances (line 10) as long as the probability of unacceptable waiting time (calculated in lines 7 and 8) is higher than a predefined threshold, p (lines 6 to 10). In our edge computing implementation of the algorithm, the calculated set of streaming engines are randomly instantiated at the edge nodes.

4.3 Network Topology

Fig. 7 demonstrates the three-tier network, which consists of the cloud, edge or CDN nodes and user groups. We generate an undirected random network topology graph using the



(a) (b) (c) (d)

Fig. 6: Visualization of the utility functions for the requirements of (a) 50 ms latency, (b) 300 Mbps bandwidth, (c) 25 requests/second processing capability, and (d) 95% availability.

Algorithm 2: Cloud-Based Scheduling

Data: Set of QoS requirements offered by the ASP: $S = \{QoS(x); x \geq 0\}$
Data: Estimated mean inter-arrival times: $\frac{1}{\lambda}$
Data: Estimated mean requests' durations: $\frac{1}{\mu}$
Result: Number of VMs to acquire: V

begin
  calculate the smallest number of V such that $\frac{\lambda}{V\mu} \leq 1$;
  $t = \min(QoS(x)); QoS(x) \in S$;
  $P(w(r_i) > t) = 1$;
  $\rho = \frac{\lambda}{V\mu}$;
  while $P(w(r_i) > t) > p$ **do**
    calculate $P_W = \frac{(V\rho)^V}{V!}((1-\rho)\sum_{n=0}^{V-1}\frac{(V\rho)^n}{n!} + \frac{(V\rho)^V}{V!})^{(-1)}$;
    calculate $P(w(r_i) > t) = P_W e^{(-V\mu(1-\rho))t}$;
    if $P(w(r_i) > t) > p$ **then**
      $V = V + 1$
  return V

BRITE topology generator (Medina et al., 2001) between the CDN and Edge nodes (shown with a dashed line in Fig. 7). We employ the Barabási–Albert scale-free network generation model (Barabási and Albert, 1999), which is known to accurately represent human-made systems such as the Internet. It mimics the incremental growth and preferential node connectivity behaviors of such systems. The nodes are added gradually and the new ones link with higher probability to well-connected nodes called the hubs.

The nodes are placed on a 25×25 km grid uniformly at random as shown with black filled circles in Fig. 8 (a). We assume that these locations are also points of interests (POIs) such as faculties in a campus or commercial buildings or recreational areas in a city. Hence, the users, represented with grey circles, are clustered around these POIs. We use Gaussian distribution to generate x and y coordinates relative to the coordinates of POIs, assign each user to the closest node, and consequently obtain user groups. Note that, the closest node may not always be the POI for which the user coordinates are generated.

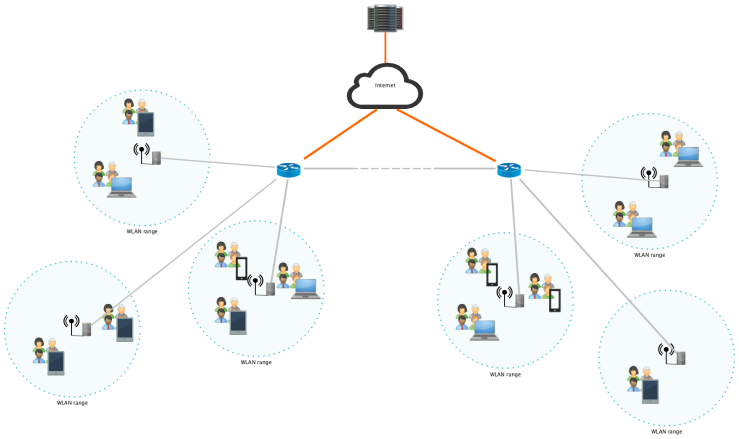


Fig. 7: Edge network infrastructure [color online].

4.4 Mobility

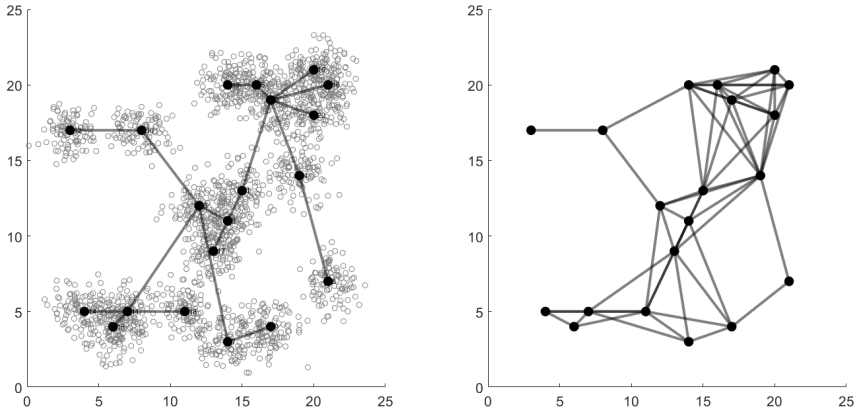
There is a large body of literature regarding human mobility based on observed movements (Hess et al., 2016). Considering the POI-based edge node placement and user distribution, we implement a mobility model that is based on real-world check-ins (Fan et al., 2016). Authors in this work analyzed a massive dataset of 37 million check-in records to understand the movement behavior of users. They determined the distribution of transition probabilities between POI's, which we utilize to represent the probability that a user leaves an edge node and joins another one.

Specifically, we consider the probabilities that a user stays in the same POI (P_{stay}), moves to a new one (P_{new}), and returns to the previous one (P_{return}). Based on the findings of Fan et al. (2016), P_{stay} is independent of time and follows a normal distribution with an average value of 0.445. P_{new}, on the other hand, decays as the power law of time, which is given in Eq. (5). Here, -0.3 is the decaying rate of P_{new}, whereas a is the rate of exploration, that is the extent that a user explores new places. Finally, P_{return} is calculated as given in Eq. (6).

$$P_{new} = a * t^{-0.3} \tag{5}$$

$$P_{return} = 1 - (P_{new} + P_{stay}) \tag{6}$$

Since the transition functions are estimated by fitting to real data, it is possible that $P_{new} + P_{stay} > 1$ in some cases. When this happens, we take $P_{new} = 1 - P_{stay}$ instead of Eq. (5). In our implementation, all POIs have the same attraction level, hence the same transition probability. However, we restrict the transitions based on geographical proximity. Fig. 8 (b) presents available mobility paths between edge nodes determined by a distance threshold of 8 units. When a user takes the action to move to a new node, the destination is randomly selected among the neighbors of the current node, except the previous one.



(a) (b)

Fig. 8: A schematic representation of edge nodes with network topology and user distribution (a), and with mobility paths (b).

4.5 Workload

Since, to the best of our knowledge, there are no publicly available data set with information about the number of viewers joining video streams, we create one based on real-world information. The main parameters considered in our scheduler/simulation are: maximum number of concurrent clients; streaming duration; user arrival rate; and required video quality.

Live video streams can be classified with respect to different parameters, such as the number and arrival process of viewers and stream duration, as reported in the analysis of live video streaming workloads carried out in (Sripanidkulchai et al., 2004). Our experiments focus on small/medium streams, the common stream type in use (in particular in social networks), which have a peak of less than 1000 concurrent clients, and are short, with a duration of less than one hour. We modeled the user arrival process with an exponential distribution, whose mean time between two arrivals ($1\backslash l$) is 5 seconds. The time each user spends watching the streaming is defined, instead, by a heavy-tailed distribution modeled as a "truncated" version of Pareto distribution characterized by a cut-off point at 40 minutes, where the tail drastically drops off (Sripanidkulchai et al., 2004).

The audience uses various end-devices to access the stream, namely smartphone, tablet and laptop, and, therefore, the video quality (i.e., resolution and bitrate) changes according to the device used and bandwidth condition. A smartphone video is encoded in standard definition in two different resolutions and bitrates, as 640x354@640Kbps and 416x234@400Kbps. A tablet video, instead, is encoded in high definition in two different formats, 1280x720 @4400kbps and 1280x720@2500kbps, having the same resolution but different bitrates. Finally, a laptop video is encoded in two different full-HD formats, as 1920x1080@3000 and 1920x1080@5000kbps. All these video parameters were retrieved from one of the most widely used commercial services for live video streaming, namely Youtube[1].

[1] https://support.google.com/youtube/answer/2853702?hl=en

Availability of Edge servers is taken from real-world failure traces for Local Domain Name Servers (LDNS) (Pang et al., 2004). This conforms to our model that Edge servers are deployed on networking hardware such as ISP point of presences. The data set contains ping probes to 62,201 LDNS servers, which are initiated at exponential intervals with a mean of 1 hour. Traces are dated between March 17 to March 24, 2004. We calculate the observed availability percentage from these traces and also account for the re-initialization period after each failure.

5 Numerical Results and Discussion

5.1 Comparison of Deployment Scenarios

The aforementioned scenarios are run 10 times with the number of mobile devices ranging from 400 to 1400. The results in Fig. 9 (a) show that the edge platform (Edge and Hybrid scenarios) reduces considerable network delay with respect to the other deployment solutions. As the workload increases, Hybrid approach tends to offload more applications to the cloud, which incurs higher average network delay. In the experiments, the delay of the Edge scenario is 4 to 5 times less than the Cloud scenario, and around half with respect to CDN.

Processing time represents another critical factor in the total service time. The results of our simulations are depicted in Fig. 9 (b), which in live streaming scenario, is mainly composed of the time to prepare the video content to be delivered. Due to the limited processing capabilities of edge nodes, they have the highest processing time and become the system bottleneck as the load increases. Essentially, in the proposed approach, where each streaming engine instance has to process all the incoming requests from the associated user device, we experience a faster raise of the processing time since the scaling possibilities are rather limited due to constrained resource capabilities of the edge nodes. Therefore, predicting in advance a suitable number of VMs represents may lead to better results. In the CDN scenario, we obtain smaller processing time and less steep rise than in the Edge scenario, since most of the processing is done in the cloud servers. We observe a sudden decrease in processing time for the Hybrid scenario when the edge nodes are overloaded and some processing is offloaded to the resource-rich cloud servers. Cloud scenario, as expected, results in the lowest processing time.

As demonstrated by the overall results in Fig. 9 (c), the Hybrid approach outperforms other scenarios in most cases. It provides similar results to the Edge scenario when the nodes are not overloaded, and addresses the problem of overloaded nodes in the edge network by dispatching the applications to the cloud. It merges the advantages of both the edge and cloud scenarios. Clearly, CDN is not the best solution for latency-sensitive applications if they also require processing power (e.g., video encoding). Yet, it is still a valid solution in other scenarios, for example, if only videos with the same characteristics (bitrate, etc) are present as in offline streaming. Cloud, as well performs poorly since the network latency is highly disruptive in the live streaming scenario.

5.2 Comparison of Scheduling Approaches

The use of Edge solutions, however, require a suitable service scheduling algorithm as shown in Fig. 10 (a). Here, both scheduling algorithms are evaluated on the edge deployment; however, edge-based scheduling outperforms the cloud-based baseline significantly

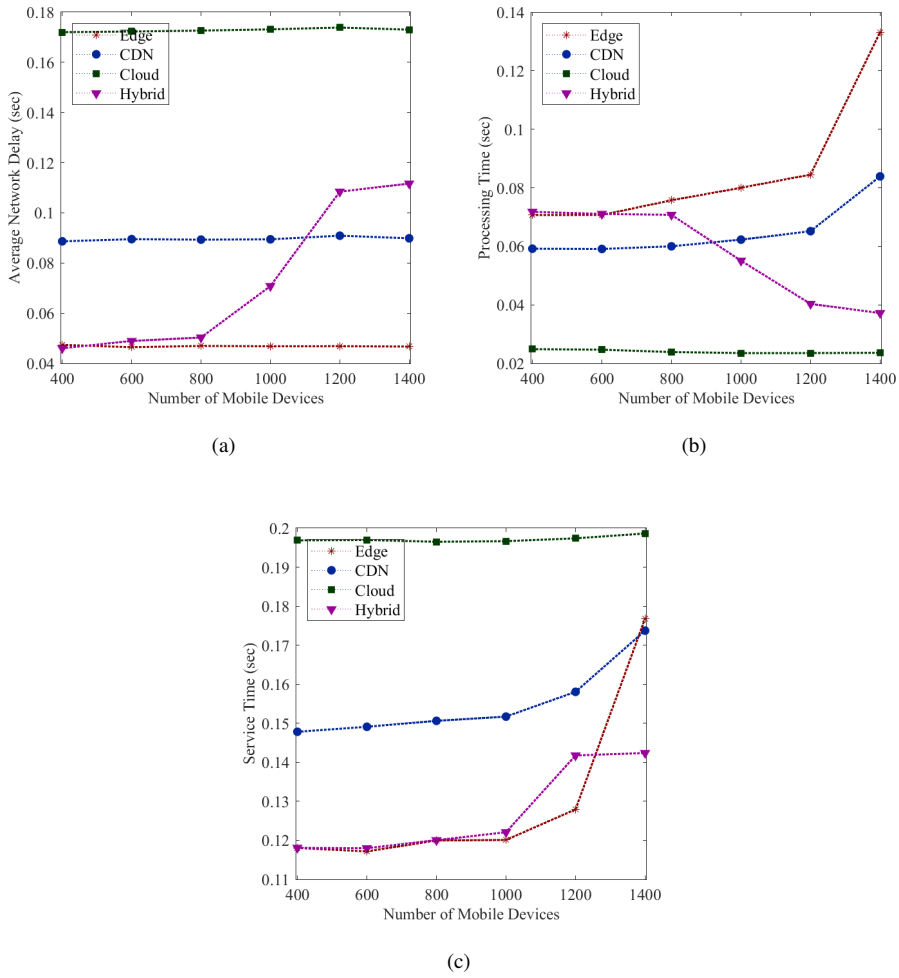


Fig. 9: Average network delay (a) average processing time (b) and average service time (c) experienced by the users in the scenario described in Sec. 4.1 [color online].

in terms of experienced network delay. This is due to the joint consideration of network conditions and user requirements. Fig. 10 (b) demonstrates that proactive estimation of the number of VM instances results in lower processing time. However, network delay improvement overshadows processing time on average and service time is shorter in nearly all cases in Fig. 10 (c). The only exception is when the edge nodes are saturated, as in the case of 1400 users. These results underline the need for scheduling solutions that take into account edge specific features to obtain optimal scheduling.

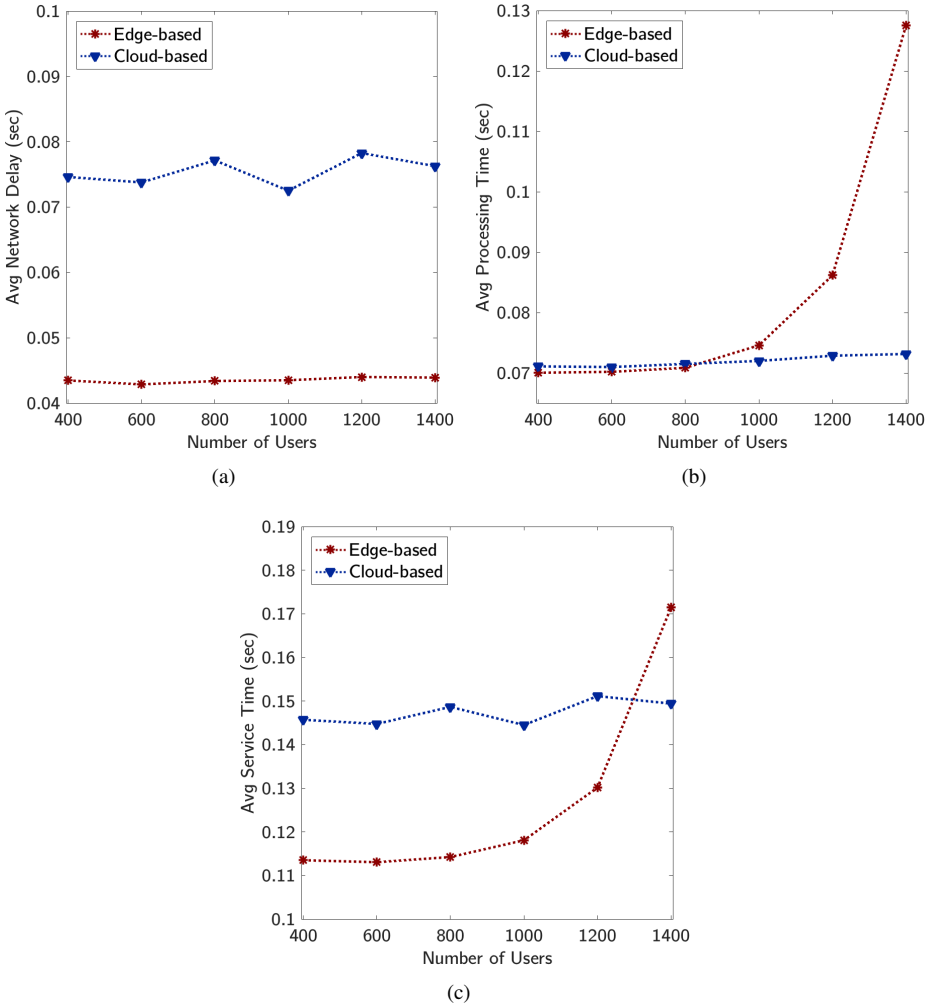


Fig. 10: Comparison of the average network delay (a) average processing time (b) and average service time (c) in the edge deployment scenario using a cloud scheduler and the edge scheduler we developed [color online].

5.3 Analysis of the Impact of User Mobility

In this final experiment, we evaluate the extent to which service quality worsens as users move over time. We particularly focus on average network delay, which is the most severely affected metric due to the increased distance between the user and a static VM placement. We present the results for the edge deployment scenario with the edge-based scheduler and 1000 users in Fig. 11. We consider five mobility scenarios from very low exploration (a = 0.05), which results in 4.47 transitions per minute, to very high exploration (a = 0.50), which results in 45.09 transitions per minute on average.

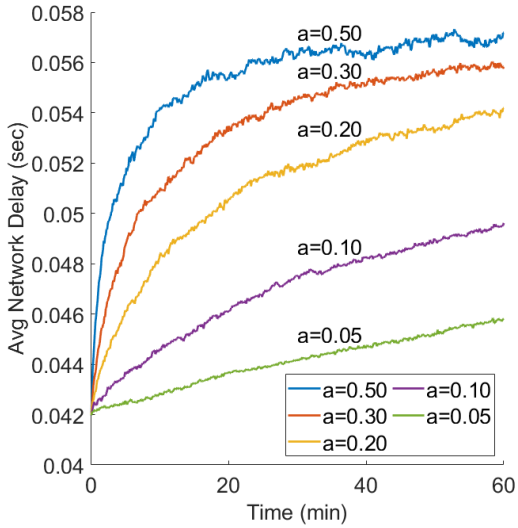


Fig. 11: Impact of user mobility on network delay with various exploration rates [color online].

Beginning from 42ms, an increase in network delay is observed for all scenarios, albeit relatively slow in low mobility scenarios. Moreover, high mobility scenarios also stabilize under 57ms of delay due to the returning behavior of the users as the time passes. Based on our analysis, the point of stabilization roughly corresponds to the time taken by every user to make at least one transition (e.g. 22:2 minutes for $a = 0.50$ or 37:5 minutes for $a = 0.30$). Despite mobility, average network delay of the proposed algorithm is significantly better than cloud-based scheduling (around 75ms) or CDN deployment (around 89ms). For the optimal performance and particularly delay-sensitive services, we recommend periodic re-execution of the scheduling algorithm, which would yield an updated mapping between resources and services. The execution time of the Alg. 1 is negligible (on the order of seconds) with respect to the re-execution period (possibly on the order of minutes or hours). However, for the environments with a very high level of mobility and with network delays between the edge nodes (e.g. vehicular services), other scheduling techniques that focus on elasticity and user mobility would be more suitable. We discuss such works from the literature in Sec. 6.

6 Related Work

Edge computing refers to the set of technologies (i.e., Mobile Edge Computing (Hu et al., 2015), Fog Computing (Mahmud et al., 2018) and Cloudlets (Verbelen et al., 2012)), which perform computation offloading and data storage at the edge of the network aiming at the reduction of end-user network delay, bandwidth consumption in core network and energy consumption (Shi et al., 2016). The development of these new technologies is mainly driven by the advent of the Internet of Things (IoT) services (e.g., real-time analytics and smart city platforms), generating a massive volume of data to be analyzed that can overwhelm current cloud-based solution and are characterized by very strict latency requirements (Dastjerdi

et al., 2016). Edge computing paradigm is proven effective in many scenarios Wang et al. (2017), however, there still exist open research challenges such as the scheduling on edge computing services. Computation offloading problem (Zhao et al., 2015; Guo et al., 2016; Mao et al., 2016), for instance, decides to schedule a task either on the mobile device or local/internet cloud. Nevertheless, there is no study that deals with the actual service scheduling on edge nodes, to the best of our knowledge. In the edge area, instead, some preliminary works have been carried out as reported by Skarlat et al. (2017) but it focuses on the optimization of response time between service components, without taking into account service users. Aazam and Huh (2015) define a resource estimation and pricing model for IoT, still without providing a scheduling approach for the service instance placement.

Preliminary evaluation and future research directions for scheduling approaches in edge scenarios are presented by Bittencourt et al. (2017). The work mainly focuses on users mobility and on the consequent elasticity requirement for service placement through three mobility-aware scheduling algorithms; it is also proposed to prioritize low delay applications in order to improve applications execution. Other works in this direction also consider handover mechanisms to cope with user mobility and unnecessary handovers via probabilistic (Zhang et al., 2017) and fuzzy logic-based (Basic et al., 2019) approaches. Sun et al. (2017), on the other hand, add energy efficiency to the equation considering the limited battery power of typical mobile users. They propose near-optimal mobility management and handover algorithms based on Lyapunov optimization. As distinct from the aforementioned works, Plachy et al. (2016) utilize mobility prediction of individual users for the proactive provisioning of VMs and communication paths. We believe mobility is less critical in the scenario considered in our work, due to the relatively short-lived nature of live video streams and wide distribution of users, both of which reduce the probability that users change groups at run-time. Additionally, since multiple users are served by a single VM, the deterioration caused by the migration of few users can be circumvented by reassigning them to other VMs by re-executing the algorithm rather than migrating the original VM.

On the other hand, the service scheduling problem is widely studied in the cloud computing context. Several approaches related to scheduling have been applied to cloud computing, e.g., (Song et al., 2014). However, they are focused on single providers and rely on a reduced number of variables. In the area of distributed clouds, Papagianni et al. (2013) propose a framework for efficient mapping of VM user requests on the aggregate set of connected clouds. They modeled the mapping problem as mixed integer programming aiming to minimize the mapping costs and the number of hops among the VMs. Again in the area of distributed clouds, Konstanteli et al. (2014) propose a novel approach to tackle the problem of service allocation in a federated cloud environment for horizontally scalable services. They define a method that allocates service component replicas taking into account the maximum service requirements in terms of computing, networking and storage resources, a set of affinity and anti-affinity rules for deploying replica in the same node or subnet and the federated infrastructure costs. They model the allocation problem as Mixed-Integer Linear Programming optimization problem and implement, then, a heuristic solver that yields to near-optimal solutions in a very short time. Pittaras et al. (2015) develop an approach for efficient mapping of virtual networks onto a multi-domain network substrate. They devise a semantic-based approach for the mapping of requests to real network subnets, which minimizes the number of hops between selected nodes. Aral and Ovatman (2016) propose a novel approach for the problem of mapping virtual networks defined by the set of interconnected VMs (i.e., service replicas) on a real infrastructure in a distributed cloud. Their solution strives to reduce network latency and optimize bandwidth utilization. It follows a topology-based mapping approach and allocates the virtual network defined by the connec-

tions among service components on a cloud subnet whose topology is isomorphic to the virtual one.

Many works are carried out also for QoS-aware service scheduling in the context of a single cloud provider, more focused, then, on the optimization of service placement within a single data center. Duong et al. (2012) propose an integrated approach that combines provisioning and scheduling techniques in order to satisfy both the provider's revenue and consumers' requirements. They devise a provisioning algorithm, based on queuing theory approaches, for the definition of the number of VMs to be deployed in order to minimize the user waiting time. Similarly, a rule-based resource provisioning algorithm that employs fuzzy logic is proposed (Jamshidi et al., 2014) to enable qualitative specification of the elasticity rules. This work also features an elasticity controller, which predicts and copes with changes in the workload. Zeng et al. (2014) develop an approach that optimizes content distribution within servers in a data center by jointly optimizing request routing and content placement. Essentially, they devise a method, which optimize the usage of storage and network capacity based on blocking probability to avoid the starvation of service users. Piao and Yan (2010), instead, present a VM placement approach for data-intensive applications that aims to minimize the transfer time between the data centers hosting the data and the VMs running the services. Therefore, they develop a solution that takes into account the transfer data time between the hosting data centers and the VMs, define a VM placement that minimizes the overall system transfer time.

However, none of these approaches can be directly applied for the VM placement in edge computing since they do not support the specific characteristics of the edge paradigm. For example, VM placement solutions for federated clouds that minimize inter-node network latency, represent only a partial solution for the VM placement for latency-sensitive applications at the edge. Indeed, these solutions are not location-aware, that is they do no take into account user localization, which is a critical feature of the edge paradigm. Definition of a VM placement that minimizes only the inter-node delay yields to sub-optimal results since the user-node latency, which accounts for a large part of the overall service delay, impedes user experience. Moreover, they are not resource-aware and therefore, considering the limited capabilities of edge nodes, the placement of a VM on edge node, which cannot provide enough resources to cope with workload peaks, incurs in high service time due to either the high processing time or the additional delay added by the high rate of VM migrations.

7 Conclusion

Effective scheduling of services in edge computing scenario is vital due to strict latency requirements and limited resources. Sub-optimal service placement and scheduling may result in significantly low quality of service and low utilization of resources. To cope with these issues, we focus on the edge service scheduling problem. We develop a service-driven approach to maximize the service quality experienced by the users through deploying services on the most suitable VMs in terms of computational and network resources.

Our proposal is a score-based algorithm that works in two stages. First, it verifies the eligibility of each available VM type, according to the service network and computational requirements as well as its reliability. It assigns a quality score to each VM type denoting the suitability of that VM to host the service to be scheduled. Then, the services are assigned in a way to maximize the total score of the chosen VMs, thus improving service quality for end users. To validate the proposed approach, we evaluated the average response and

processing time as well as network delay experienced by the users in the Edge, CDN, Cloud, and Hybrid (Edge and Cloud) scenarios. The results obtained demonstrate the validity of our scheduling approach in the scope of edge computing paradigm. Indeed, the promised benefits of edge computing can only be achieved when effective scheduling algorithms that consider its peculiar features are implemented. We believe that our work will bring more industry and research attention to the barriers to the adoption of edge computing.

As future work, we will enhance the developed solution by decentralizing the optimization process, adding another decision output for the number of instances of services, and considering vertical and horizontal scaling. Moreover, we plan to investigate and integrate our solution with Software Defined Networks to improve and cope with the latency and bandwidth requirements of edge applications.

Acknowledgements This work has been partially funded by the Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015, by the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 838949, and by the Italian National Interuniversity Consortium for Informatics (CINI).

References

Aazam M, Huh EN (2015) Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In: IEEE International Conference on Advanced Information Networking and Applications (AINA), IEEE, pp 687–694

Aral A, Brandic I (2017) Quality of service channelling for latency sensitive edge applications. In: IEEE International Conference on Edge Computing (EDGE), IEEE, pp 166–173

Aral A, Brandic I (2018) Dependency mining for service resilience at the edge. In: ACM/IEEE Symposium on Edge Computing, ACM, pp 228–242

Aral A, Ovatman T (2016) Network-aware embedding of virtual machine clusters onto federated cloud infrastructure. Journal of Systems and Software 120:89–104

Aral A, Ovatman T (2018) A decentralized replica placement algorithm for edge computing. IEEE Transactions on Network and Service Management 15(2):516–529

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Basic F, Aral A, Brandic I (2019) Fuzzy handoff control in edge offloading. In: IEEE International Conference on Fog Computing, IEEE

Bilal K, Erbad A (2017) Edge computing for interactive media and video streaming. In: International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, pp 68–73

Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M (2017) Mobility-aware application scheduling in fog computing. IEEE Cloud Computing 4(2):26–35

Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience 41(1):23–50

Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R (2016) Fog computing: Principles, architectures, and applications. In: Internet of Things, Elsevier, pp 61–75

Dobrian F, Sekar V, Awan A, Stoica I, Joseph D, Ganjam A, Zhan J, Zhang H (2011) Understanding the impact of video quality on user engagement. ACM SIGCOMM Computer Communication Review 41(4):362–373

Duong TNB, Li X, Goh RSM, Tang X, Cai W (2012) Qos-aware revenue-cost optimization for latency-sensitive services in iaas clouds. In: IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT), IEEE, pp 11–18

Fan C, Huang J, Yang D, Rong Z (2016) Modeling poi transition network of human mobility. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, IEEE, pp 364–367

Guo X, Singh R, Zhao T, Niu Z (2016) An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems. In: IEEE International Conference on Communications (ICC), IEEE, pp 1–7

Hess A, Hummel KA, Gansterer WN, Haring G (2016) Data-driven human mobility modeling: a survey and engineering guidance for mobile networking. ACM Computing Surveys (CSUR) 48(3):38

Hu W, Gao Y, Ha K, Wang J, Amos B, Chen Z, Pillai P, Satyanarayanan M (2016) Quantifying the impact of edge computing on mobile applications. In: Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems, ACM, p 5

Hu YC, Patel M, Sabella D, Sprecher N, Young V (2015) Mobile edge computing—a key technology towards 5g. ETSI White Paper 11(11):1–16

Jamshidi P, Ahmad A, Pahl C (2014) Autonomic resource provisioning for cloud-based software. In: International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM, pp 95–104

Konstanteli K, Cucinotta T, Psychas K, Varvarigou TA (2014) Elastic admission control for federated cloud services. IEEE Transactions on Cloud Computing 2(3):348–361

Mahmud R, Kotagiri R, Buyya R (2018) Fog computing: A taxonomy, survey and future directions. In: Internet of everything, Springer, pp 103–130

Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications 34(12):3590–3605

Medina A, Lakhina A, Matta I, Byers J (2001) BRITE: An approach to universal topology generation. In: Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, pp 346–353

Pang J, Hendricks J, Akella A, De Prisco R, Maggs B, Seshan S (2004) Availability, usage, and deployment characteristics of the domain name system. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM, pp 1–14

Papagianni C, Leivadeas A, Papavassiliou S, Maglaris V, Cervello-Pastor C, Monje A (2013) On the optimal allocation of virtual resources in cloud computing networks. IEEE Transactions on Computers 62(6):1060–1071

Pathan AMK, Buyya R (2007) A taxonomy and survey of content delivery networks. Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report 4

Piao JT, Yan J (2010) A network-aware virtual machine placement and migration approach in cloud computing. In: International Conference on Grid and Cooperative Computing (GCC), IEEE, pp 87–92

Pittaras C, Papagianni C, Leivadeas A, Grosso P, van der Ham J, Papavassiliou S (2015) Resource discovery and allocation for federated virtualized infrastructures. Future Generation Computer Systems 42:55–63

Plachy J, Becvar Z, Strinati EC (2016) Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In: IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications, IEEE, pp 1–6

Satyanarayanan M (2017) The emergence of edge computing. Computer 50(1):30–39

Scoca V, Uriarte RB, De Nicola R (2017) Smart contract negotiation in cloud computing. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), IEEE, pp 592–599

Scoca V, Aral A, Brandic I, De Nicola R, Uriarte RB (2018) Scheduling latency-sensitive applications in edge computing. In: CLOSER, pp 158–168

Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet of Things Journal 3(5):637–646

Skarlat O, Nardelli M, Schulte S, Dustdar S (2017) Towards qos-aware fog service placement. In: IEEE 1st International Conference on Fog and Edge Computing (ICFEC), IEEE, pp 89–96

Song W, Xiao Z, Chen Q, Luo H (2014) Adaptive resource provisioning for the cloud using online bin packing. IEEE Transactions on Computers 63(11):2647–2660

Sonmez C, Ozgovde A, Ersoy C (2017) Edgecloudsim: An environment for performance evaluation of edge computing systems. In: International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, pp 39–44

Sripanidkulchai K, Maggs B, Zhang H (2004) An analysis of live streaming workloads on the internet. In: ACM SIGCOMM Conference on Internet Measurement, ACM, pp 41–54

Stanciu A (2017) Blockchain based distributed control system for edge computing. In: 2017 21st International Conference on Control Systems and Computer Science (CSCS), IEEE, pp 667–671

Sun Y, Zhou S, Xu J (2017) EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE Journal on Selected Areas in Communications 35(11):2637–2646

Tuli S, Mahmud R, Tuli S, Buyya R (2019) Fogbus: A blockchain-based lightweight framework for edge and fog computing. Journal of Systems and Software

Uriarte RB, De Nicola R (2018) Blockchain-based decentralised cloud/fog solutions: Challenges, opportunities and standards. IEEE Communications Standards Magazine

Uriarte RB, Tiezzi F, De Nicola R (2016a) Dynamic slas for clouds. In: European Conference on Service-Oriented and Cloud Computing, Springer, pp 34–49

Uriarte RB, Tiezzi F, Tsaftaris SA (2016b) Supporting autonomic management of clouds: Service clustering with random forest. IEEE Transactions on Network and Service Management 13(3):595–607

Uriarte RB, De Nicola R, Scoca V, Tiezzi F (2019) Defining and guaranteeing dynamic service levels in clouds. Future Generation Computer Systems

Verbelen T, Simoens P, De Turck F, Dhoedt B (2012) Cloudlets: Bringing the cloud to the mobile user. In: ACM workshop on Mobile cloud computing and services, ACM, pp 29–36

Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W (2017) A survey on mobile edge networks: Convergence of computing, caching and communications. IEEE Access 5:6757–6779

Yu C, Lumezanu C, Sharma A, Xu Q, Jiang G, Madhyastha HV (2015) Software-defined latency monitoring in data center networks. In: International Conference on Passive and Active Network Measurement, Springer, pp 360–372

Zeng L, Veeravalli B, Wei Q (2014) Space4time: Optimization latency-sensitive content service in cloud. Journal of Network and Computer Applications 41:358–368

Zhang H, Qiu Y, Chu X, Long K, Leung VC (2017) Fog radio access networks: Mobility management, interference mitigation, and resource optimization. IEEE Wireless Communications 24(6):120–127

Zhao T, Zhou S, Guo X, Zhao Y, Niu Z (2015) A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. In: IEEE Globecom Workshops, IEEE, pp 1–6